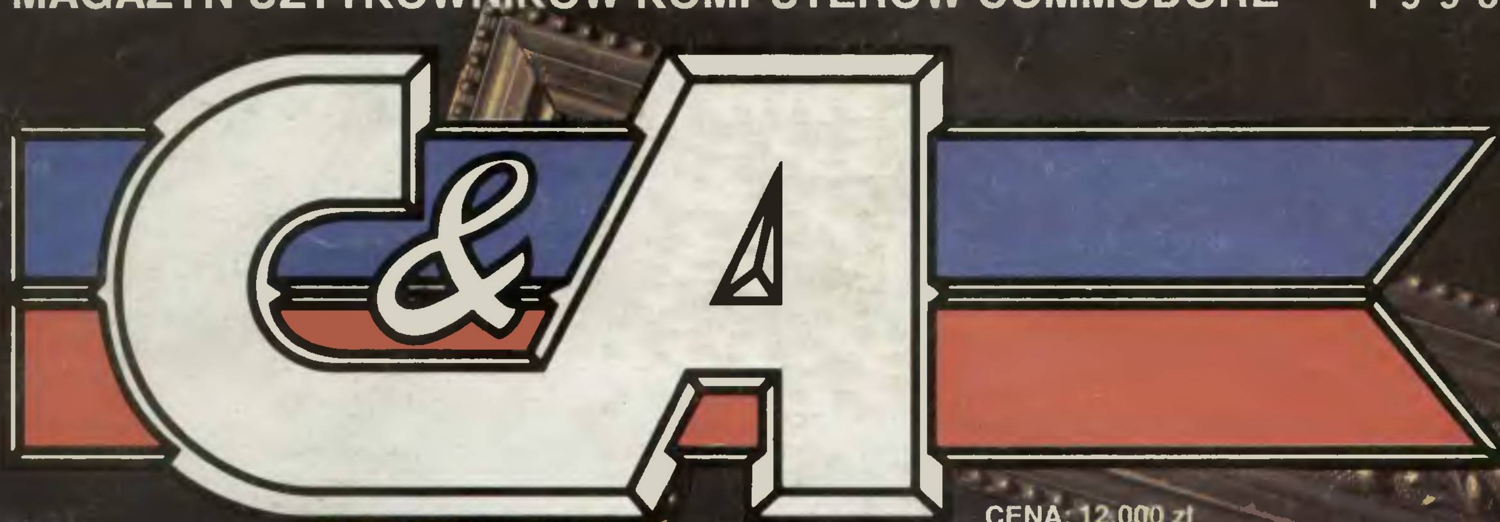
Commodore & Amiga nr1

styczeń 1993

MAGAZYN UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE



NR INDEKSU 355216
ISSN 0867-8022

CENA: 12.000 zł

AMIGA

DEMOSY (cz. II)

JAK I NA CZYM UZYSKAĆ OBRAZ Z AMIGI

C-64

JAK POŁĄCZYĆ DWA C-64?

CO TO SĄ SPRAJTY?

LIFE IS LIFE – GRA W PROGRAMOTECE!

DEMOSY (cz. 2)

Megademo II / Cryptoburners

PERŁY Z LAMUSA, CZYLI O LEGENDACH SCENY

W grudniowym numerze opisywałem wszystko ogólnie, abyście szybko zorientowali się w temacie. Niemniej wiecie już sporo: kto robi demka, kto to taki jest koder, a kto sysop. Jednak prawdziwemu maniakowi sceny nigdy nie wystarczą same słownikowe definicje (bo te „zawody" powinny się już dawno znaleźć w słowniku slangu komputerowego). Każdy fan demosów powinien być zawsze „na bieżąco" i wiedzieć wszystko, co wydarzyło się do tej pory na światowej i polskiej scenie. Tak więc w tym artykuliku postaram się opisać mniej więcej ewolucję „demosów" aż po rok 1990. Jakie grupy w przeszłości były „na topie" i jakie demka były w swoim czasie najlepsze. No to zaczynamy.

Pierwsze amigowskie grupy zaczęły powstawać już w 1987 roku (z tego, co wiem, już wtedy istniały na scenie grupy będące dzisiaj sławami, jak ne przykład PHENOMENA, THE SILENTS, KEFRENS czy FAIRLIGHT). Demosy były wówczas na ogół plikowymi pokazami jednego efektu (zazwyczaj na gorszym poziomie niż obecne intra renomowanych grup). Efekty stanowiły na ogół różnego rodzaju sinus-*scrolle* (*scroll* jest to, jak wiecie, najpopularniejszy element dem, polegający na płynnym przesuwaniu (ang. scrolling) napisów; sinus - to nic innego, jak dodane pofałdowanie, najczęściej w kształt sinusoidy), „ploty" (są to znane z gorszych komputerów animowane w czasie rzeczywistym wzorki utworzone z punktów), e także „druciena", czyli niewypełniana wektorówka. Grafika wektorowa to chyba w obecnych czasach to, z czym przede wszystkim kojarzą się nam demosy, więc na chwilę zatrzymamy się przy niej.

Wektorki występowały w demkach od niepamiętnych czasów (może trochę przesadziłem, ale z pewnością pojawiły się już w 1987 roku, czyli zaraz po tym, kiedy firma Commodore wypuściła na rynek Amigę), jednak prawdopodobnie nikt nie zwracał na ten efekt specjalnej uwagi aż do momentu, kiedy na targach Cebit'90 zaprezentowała swoje demo grupa RED SECTOR. Nie były to pierwsze wypełniane wektory, nie były to nawet pierwsze „niewypukłe" wektory, za to było to pierwsze perfekcyjnie dopracowane demo z wykorzystaniem tego typu grafiki. Nie raz już okazywało się, że sławny zostaje nie ten koder, który wymyśli dany efekt, tylko ten, który potrafi go dopracować i „dobrze sprzedać".

Wróćmy jednak do grafiki wektorowej i do tego, dlaczego jest ona tak popularna. Na czym polega urok wektorówki? Być może wielu specjalistów nie zgodzi się ze mną, mam jednak własną teorię dotyczącą wektor-manii. Otóż, jak wiadomo, prawie każdy chciałby oglądać w telewizji obraz trójwymiarowy. Niejednokrotnie mamy ochotę spojrzeć na zdjęcie od drugiej strony (czyż nie?), ale każdy dobrze wie, że przy obecnym poziomie techniki jeszcze przez jakiś czas nie będzie to możliwe. Jeżeli chcielibyśmy na przykład zobaczyć, jak jakiś człowieczek odwraca się do nes tyłem, to musielibyśmy narysować animację klatka po klatce (jak w przypadku filmu animowanego). Jeżeli chcemy, aby ów człowieczek odwracał się w lewą stronę albo do góry nogami, to musimy dorysować kolejne klatki. A gdyby tak móc zapamiętać definicję tego człowieczke, dzięki czemu będzie go potem można oglądać ze wszystkich stron? To marzenie spędzało zapewne sen z powiek

COMMODORE & AMIGA 1/93

Tym razem serwujemy Wam, drodzy Czytelnicy, sporą porcję „komputerowej kuchni", powiem więcej: cały styczniowy numer „C&A" przesycony jest takąż tematyką. I tak użytkownikom Amigi proponujemy aż cztery obszerne artykuły, w których autorzy próbują odpowiedzieć (mam nadzieję, że im się to uda) na nieśmiertelne pytania „jak?" i „dlaczego?". Opracowanie pt. „Jak i na czym uzyskać obraz z Amigi?" z pewnością pomoże każdemu, kto dopiero co kupił Amigę i zastanawia się właśnie, jak by ją podłączyć do stojącego w mieszkaniu telewizora lub starego, na pierwszy rzut oka bezużytecznego monitora. Kolejne dwa artykuły odsłaniają tajniki budowy i działania podstawowych układów Amigi: Complex Interface Adapter (CIA) i procesora Motorola 68000. Moim skromnym zdaniem z tych oraz z następnego, pokaźnego rozmiarami, artykułu można się wiele nauczyć, zwłaszcza, jeśli ktoś myśli poważnie o samodzielnym programowaniu - bez podstawowej wiedzy o funkcjonowaniu istotnych układów i ich rejestrów nie ma co marzyć o uzyskaniu efektów, o których na stronie obok pisze nasz niezrównany korespondent z demosceny, NINJA/ACTION DIRECT.

Do działu „komputerowej kuchni" należy też zaliczyć piąty już odcinek o programowaniu w AMOSie, obecnie bardzo popularnym języku dla Amigi. Także i do tej lektury serdecznie zapraszam.

Jeśli chodzi o C-64, to polecam dwa bardzo ciekawe opracowania Bartka Kachniarza: „Z czym jada się logo?" oraz „Co to są sprajty?" - dla programistów rzecz nieoceniona. Namawiam też do przeczytania artykułu poruszającego odwieczny problem: co lepsze, „małe" ATARI czy C-64? Tym razem wypowiada się na ten temat zagorzały atarowiec - warto poznać jego zdanie.

Kontynuujemy oczywiście stałe cykle: Asembler 6502, Jak napisać własne demo (proszę, piszcie, czy ten cykl się Wam podoba, czy spełnia Wasze oczekiwania?) i - tu ukłon w stronę posiadaczy C-128/16/116 i pochodnych - Grafika dla każdego (cz. 2).

Poza tym nasz hardware'owiec, Jurek Dudek, rozpoczyna prowadzenie stałego działu „Zrób to sam". Początek skromny: połączenie dwóch C-64, lecz już od następnego numeru coś, co powinno niejednego zainteresować: sterowanie urządzeń zewnętrznych poprzez USER PORT.

Na koniec zapraszam do przeczytania jeszcze gorącej relacji z targów we Frakfurcie nad Menem (odbywały się one w dniach 27-29 listopada 1992), gdzie firma Commodore zaprezentowała swoją najnowszą... ale o tym przeczytajcie już sami.

CHRISTIAN GRZENKOWICZ

AMIGA

C 64

C-16/116/+4/128

ORAZ

Redaktor naczelny: Klaudiusz Dybowski
Z-ca red. naczelnego: Christian Grzenkowicz
Zespół redakcyjny: Andrzej Bobek (szef działu Amigi), Robert Chojecki, Dariusz Ducki
Opracowanie graficzne: Studio Linea i Jolanta Przeździecka
Zdjęcia: Jerzy Stokowski

Stali współpracownicy: Rafał Borzyński, Jerzy Dudek, Piotr Cerkiewnik, Bartłomiej Dramczyk, Mariusz Ferdyn, Paweł Galas, Bartłomiej Kachniarz, Wojciech Kazimierczak, Robert Kuliś, Piotr Liszewski, Rafał Piasek, Olaf Przybyszewski, Bartosz Smaga

Redakcja: ul. Wasilkowskiego 7, 02-776 Warszawa, tel. 643-18-40
Kontakt z Czytelnikami: piątek w godzinach 13.00-17.00
Wydawca: Spółdzielnia „Bajtek", ul. Wspólna 61, 00-687 Warszawa, tel./fax 21-12-05

Druk: Przedsiębiorstwo Wydawniczo-Poligraficzne „GRYF", Sp. Akc. Ciechanów

JAK I NA CZYM
UZYSKAĆ OBRAZ Z AMIGI ?

Jeśli masz problemy z przyłączeniem swojej Amigi do telewizora czy monitora, którym aktualnie dysponujesz, koniecznie przeczytaj ten artykuł. Dowiesz się z niego, w jaki sposób można podłączyć do Amigi różne nietypowe odbiorniki, poznasz też ich możliwości. Pięć opisanych tu urządzeń reprezentuje właściwie większość dostępnych na rynku - różnice dotyczą co najwyżej nazwy.



Monitor Commodore 1084S

Monitor to bezsprzecznie jeden z najważniejszych elementów zestawu komputerowego. Z jednej bowiem strony umożliwia on nam komunikację z komputerem, z drugiej - od jego jakości zależy przecież wpływ pracy z komputerem na nasze zdrowie i samopoczucie. Jeżeli dysponujemy naprawdę wysokiej jakości monitorem, to nawet po wielu godzinach spędzonych przy klawiaturze nie będziemy czuć specjalnego zmęczenia. Jeśli jednak skazani jesteśmy na monitor niskiej jakości, albo, co gorzej, na telewizor podłączony przez modulator, to już po godzinie pracy bardzo trudno byłoby nam się zgodzić ze stwierdzeniem, że czujemy się wypoczęci.

Amigę można podłączyć na wiele sposobów do jeszcze większej liczby wszelakiej maści monitorów i telewizorów. Jako że ze zrozumiałych względów nie jesteśmy w stanie opisać sposobów podłączenia Amigi ani też wrażeń z pracy z choćby kilkunastoma procentami dostępnych na rynku monitorów, postanowiliśmy wybrać kilka urządzeń w miarę dobrze reprezentujących sprzęt, który można do Amigi w charakterze monitora podłączyć.

Sanyo CED 3011PV

Jest to dość popularny telewizor z kineskopem 14-calowym, dość często stosowany do współpracy z komputerami domowymi. Przyłączenie do niego Amigi jest możliwe bezpośrednio (tylko obraz monochromatyczny) lub za pośrednictwem modulatora. Telewizor ten ma tylko złącza Cinch, co wyklucza bezpośrednią współpracę z Amigą w kolorze (bezpośrednią, czyli bez modulatora, a więc w trybie RGB).

Uzyskiwany obraz (oczywiście po podłączeniu Amigi po tzw. niskiej częstotliwości, czyli do gniazd Cinch, a nie za pośrednictwem gniazda antenowego, gwarantującego najgorszą jakość obrazu) jest niezły i idealnie nadaje się na przykład do gier. W trybach tekstowych czytelność jest w miarę przyzwoita, lecz w trybie Hi-Res Interlace trzeba się już często domyślać, co właściwie jest na ekranie wyświetlane.

Zaletą tego telewizora jest umiarkowane drżenie obrazu w trybie Interlace. Do wad należy zaliczyć bardzo kiepskie wyświetlanie napisów w ciemnych odcieniach koloru niebieskiego. Są one rozmyte, co niesłychanie utrudnia czytelność.

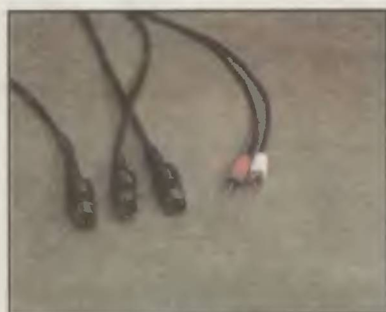
Nie jest możliwa współpraca z Amigą w systemie NTSC, gdyż telewizor ten nie utrzymuje synchronizacji pionowej 60 Hz i na ekranie widzimy jakby cztery obrazy przesunięte względem siebie w pionie, w dodatku drżące. Istnieje jednak odmiana tego telewizora umożliwiająca pracę w systemie NTSC i oznaczona jest ona CEM 6011PV.

Dźwięk uzyskiwany za pośrednictwem telewizora jest monofoniczny i sugerowalibyśmy raczej podłączenie Amigi do zewnętrznego wzmacniacza.

Podsumowując, należy stwierdzić, że telewizor ten doskonale nadaje się dla amatorów gier. Co do programów użytkowych, to jeśli użytkownik jest gotów pogodzić się z bardzo niskim komfortem pracy, również może zdecydować się na zastosowanie omawianej konfiguracji. Pamiętajcie jednak, że telewizor podłączony do Amigi przez modulator to rozwiązanie dające najniższą jakość obrazu, dobre może dla fanów gier komputerowych, absolutnie jednak wykluczające możliwość wykorzystania Amigi do zastosowań poważniejszych. Kilka godzin pracy z edytorem tekstów używającym trybu osiemdziesięciu kolumn wywrze na Was z pewnością niemiłe wrażenie.

Westa 401

Nie zawsze jednak stojący w domu telewizor musi dawać obraz bardzo niskiej jakości. Coraz więcej bowiem produkowanych na świecie telewizo-



wtyki typu CINCH



wtyki typu SCART



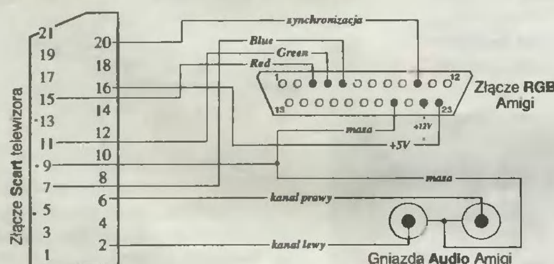
wtyki typu RGB

rów wyposażanych jest w wejście RGB, umożliwiające doprowadzenie danych o obrazie w tej właśnie formie. Gwarantuje to najwyższą jakość, jaką mogą zapewnić układy i kineskop telewizora, ponieważ przy przesyłaniu danych w formie Composite Video wszystkie synały zakłócają się nawzajem i tylko ich odseparowanie (jak przy RGB) pozwala na uzyskanie dobrej jakości obrazu.

Tak więc uznaliśmy, że nie od rzeczy będzie wspomnieć o telewizorze wyposażonym w wejście Euro-Scart, dające możliwość doprowadzenia sygnałów RGB. Jako przykład wybraliśmy sprzęt polskiej produkcji, telewizor Westa 401, będący na przyzwoitym poziomie technologicznym, a przede wszystkim mający zachodni (importowany) kineskop.

Jakość obrazu serwowanego przez Westę podłączoną do Amigi przez modulator nie odbiega co prawda specjalnie od normy, gdy jednak podłączymy ją wykorzystując sygnały RGB, obraz będzie o niebo lepszy. Jeśli telewizor jest dobrze wyregulowany i, co najważniejsze, ma dobry kineskop, to obraz uzyskiwany z jego pomocą może być nawet porównywalny z tym, który możemy oglądać na firmowym monitorze Commodore 1084S. Często jednak, na przykład w mniejszych Westach oznaczonych symbolem 201, kineskop może być kiepskiej jakości i obraz będzie pozostawiał bardzo wiele do życzenia.

Jeśli więc planujecie zakup telewizora, który chcecie wykorzystywać także jako monitor, radzilibyśmy dobrze i starannie wybierać, a przede wszystkim trzymać się z daleka od telewizorów z tragicznej wprost jakości polskimi kineskopami.



Uwaga: w przypadku monitora nie potrzeba doprowadzać napięcia przełączającego +5V
*Uwaga2: jeżeli telewizorowi nie wystarcza napięcie +5V, można spróbować **ostrożnie i na** własne ryzyko doprowadzić +12V*

RYS. 1 Schemat połączenia telewizora ze złączem typu EURO–SCART

Monitor Neptun 156

Nie wszystko, co polskie, od razu musi być złe. Do niczego są nasze rodzime kineskopy kolorowe, całkiem za to przyzwoite są te monochromatyczne, co zresztą jest jedynie konsekwencją faktu, że wymagają one prymitywniejszej technologii. W każdym bądź razie ze spokojnym sumieniem możemy polecić Wam polski monitor monochromatyczny Neptun 156. Jakość uzyskiwanego z jego użyciem obrazu jest naprawdę dobra, lepsza, niż na firmowych monitorach Commodore lub Philips, przełączonych w tryb monochromatyczny.

Najczęściej spotykane wady to złe wyregulowanie opisywanego monitora, objawiające się zniekształceniami obrazu. W takim przypadku wystarczy jednak drobna korekta ustawień potencjometrów dostępnych w tylnej części monitora. Niektóre egzemplarze mają też tendencję do generowania głośnego pisku o częstotliwości 15 kHz.

Neptun 156 jest, w naszej opinii, dobrą propozycją dla tych, którzy nie przywiązują specjalnej wagi do koloru. Zresztą jego niska cena czyni go doskonałym rozwiązaniem na czas, jaki musi niekiedy upłynąć, zanim niektórych z Was będzie po kupnie Amigi stać na monitor kolorowy.

Commodore 1084S

Jest to firmowy monitor Commodore, reklamowany jako skonstruowany specjalnie dla Amigi. Postawione przed nim zadanie spełnia bardzo dobrze. Ma cztery rodzaje wejść, co umożliwia współpracę z wieloma komputerami - i nie tylko. Pierwsze z nich, Analog RGB, najlepiej nadaje się do współpracy z Amigą czy Atari ST. Drugie - Digital RGB (cyfrowe wejście RGB) - umożliwia współpracę z komputerami PC (karta CGA), na przykład Commodore PC-10/20. Trzecie gniazdo jest standardowym wejściem Composite Video typu Cinch, umożliwiającym podłączenie m.in. magnetowidu lub tunera TV Sat, a także komputerów C-64, ZX Spectrum czy też „małego" Atari. Jednak do współpracy z C-64 lepiej nadaje się wejście Luma/Chroma (odseparowane sygnały luminancji i chrominancji). Dzięki oddzieleniu sygnału chrominancji od luminancji ten drugi nie jest zakłócany przez pierwszy, co ma miejsce w zespolonym sygnale wizji (Composite Video).

Przejdźmy do jakości obrazu. Ekran jest wyposażony w specjalną warstwę antyodbiciową, co znacznie poprawia czytelność obrazu w jasnych pomieszczeniach (światło z innych niż kineskop źródeł jest w dużej mierze pochłaniane). Sam obraz jest bardzo czytelny, a kolory naturalne i „żywe". Czytelność liter jest również bardzo dobra w trybie Hi-Res Interlace, choć drżenie obrazu jest dość dokuczliwe, szczególnie przy kontrastowych kolorach tła i liter. Monitor pracuje poprawnie także w systemie NTSC, widoczne jest nieznaczne zwężenie obrazu w pionie (można je skorygować potencjometrem znajdującym się z tyłu monitora, lecz wtedy po przejściu do systemu PAL konieczna będzie ponowna korekta ze względu na to, że nie będą widoczne dolne i górne linie obrazu). Przydatny jest też przełącznik GREEN/COLOR umożliwiający przejście do trybu pracy monochromatycznej w kolorze zielonym. Przy długiej pracy, na przykład z edytorem tekstu, unikamy zmęczenia oczu.

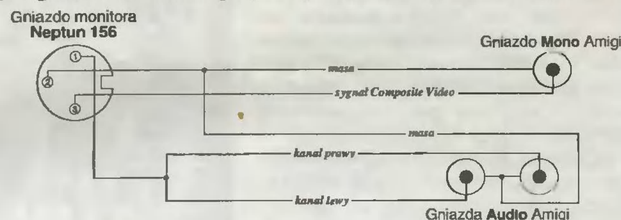
Od strony dźwięku niewiele można monitorowi zarzucić. Wyposażony jest w dwa głośniki (stereo) o mocy 1 W każdy. Daje to nam całkiem niezłą jakość dźwięku w grach i programach demonstracyjnych. Do programów muzycznych bardzo pożądane byłoby jednak przyłączenie komputera do wzmacniacza i zewnętrznych kolumn głośnikowych.

Wady opisywanego monitora to duża głębokość (długość), bardzo utrudniająca jego ustawienie, i przede wszystkim wysoka cena. Ogólnie monitor można określić jako doskonały do wszelkich zastosowań Amigi.

Rainbow SVGA

Z całą odpowiedzialnością można jednak stwierdzić, że żadne z omówionych do tej pory rozwiązań nie nadaje się dla ludzi używających Amigi profesjonalnie, do ciężkiej pracy. Dla takich zastosowań potrzebny jest monitor naprawdę najwyższej klasy, gwarantujący bezbłędną ostrość obrazu, umożliwiający wyeliminowanie migotania, a także pozbawiony takich wad, jak na przykład generowanie szkodliwego dla zdrowia i niesłychanie męczącego pisku o częstotliwości 15 kHz.

Za spełniający te wymagania uznaliśmy koreański monitor Rainbow SVGA. Przeznaczony jest on z założenia do współpracy z „pecetami", można go jednak podłączyć do niektórych Amig. Wymaga on sygnału o częstotliwości 31 kHz, co w praktyce oznacza tyle, że poprawnie będzie pracował tylko z Amigami wyposażonymi w tzw. flicker-fixer, czyli kartę likwidującą migotanie obrazu. Z góry oczywiście drastycznie zawęża to krąg



RYS. 2 Schemat połączenie telewizora ze złączem typu DIN

zainteresowanych, nic w tym jednak dziwnego, w końcu Amigi używa się raczej do zastosowań domowych. Niemniej jednak poważniejsza praca możliwa jest dopiero po wyeliminowaniu migotania, a więc po podwojeniu częstotliwości wyświetlania do 31 kHz. Jeśli już więc wydało się kilka milionów na flicker-fixer, nie ma sensu oszczędzać na monitorze, tym bardziej, że dobry, „pecetowy" monitor nie jest wcale droższy od typowo „amigowych" oferowanych na giełdzie. Oczywiście trzeba też liczyć koszt flicker-fixera, jednak różnica w jakości jest po prostu ogromna.

Należy naturalnie pamiętać, że nie wszyskie monitory do „pecetów" są dobre. Często montowane w nich kineskopy są dyskusyjnej jakości, regułą jest zresztą, że dopiero monitor reklamowany jako SVGA (czyli umożliwiający wyświetlenie w linii około 1000 punktów) pozwoli na uzyskanie naprawdę dobrej jakości obrazu w trybie Hi-Res Interlace, czyli 640x512 punktów.

Kupując „pecetowy" monitor należy upewnić się, czy istnieje możliwość jego zwrotu w terminie kilku dni, nie zawsze można bowiem wszystko sprawdzić w sklepie. No i przede wszystkim należy z dystansem podchodzić do monitorów marki „Monitor" - sytuacja jest dokładnie identyczna, jak na całym rynku sprzętu „pecetowego".

W każdym razie wybrany przez nas na reprezentanta całej grupy Rainbow SVGA spisuje się doskonale i absolutnie nic nie można mu zarzucić. Niestety, w żaden sposób nie da się go zmusić do pracy ze zwykłą Amigą.

ANDRZEJ BOBEK
JERZY DUDEK

DEMOSY (cz. 2)

ciąg dalszy ze str. 2

wielu programistom (pewnie jeszcze wtedy, gdy my jeszcze nie wiedzieliśmy, co to komputer), którzy za wszelką cenę chcieli stworzyć algorytm umożliwiający oglądanie ze wszystkich stron trójwymiarowych obiektów na dwuwymiarowym monitorze. Przydałoby się również, żeby projekty, które dotąd rysowano w trzech rzutach na jednym ekranie, można było „obracać" wokół trzech osi. No i jakiś bardzo mądry człowiek wymyślił, że każdy rysunek można sprowadzić do wektorowej figury geometrycznej (oczywiście, im definicja będzie dokładniejsza, tym więcej obliczeń musi wykonywać komputer).

Tak się zaczęło i dziś grafika wektorowa w amigowych demach jest już chyba bliska ideału. Cały czas jednak powstają coraz to nowsze demka, w których obiekty są coraz bardziej skomplikowane, a procedury obrotów coraz to szybsze. Zaczynało się od prostych sześcianów, a dziś w demie SOUND VISION/REFLECT możemy już podziwiać wektorowe miasteczko (!) z latającym nad nim źródłem światła, oświetlającym domy z różnych stron. Co będzie za rok? Zobaczymy.

POCZĄTKI

Teraz czas na obiecane perły z lamusa, czyli demka, które przeszły do historii ze względu na nowe idee (pomysły, nowe rozwiązania) i, miejmy nadzieję, nie zostaną jeszcze długo zapomniane.

Na początek jedno z pierwszych dobrych dem całodyskowych, a mianowicie MEGADEMO grupy WILD COPPER. Było ono raczej nietypowe i nawet do dziś jest jedyne w swoim rodzaju. Jego części wybierało się bowiem z Workbench-a! Nie był to jednak normalny Workbench i w owym czasie wywoływał on ogromne zainteresowane. Nie dość bowiem, że cały czas grała muzyka (wtedy multitasking (z ang. wielozadaniowość) nie był jeszcze tak powszechnie wykorzystywany jak obecnie), to jeszcze na dole ekranu leciał śliczny sinusoidalny *scroll*, a kursor myszki kręcił się w rytm muzyki. Same części demka w owym czasie również były uważane za fantastyczne, dzięki znakomitym „drucianym" wektorom czy też dzięki *scrollowi* opisanemu na okręgu.

Mniej więcej w tym samym czasie wydała swoje pierwsze MEGADEMO grupa SCOOPEX. Nie było w nim jeszcze żadnych wektorów, ale za to zostało ono dobrze dopracowane, a procedurki stały na wysokim poziomie (oczywiście mówimy o poziomie w roku 1989).

W tym samym roku bardzo aktywna była już grupa KEFRENS, która do momentu wydania pierwszego „trackma" (MENTAL HANGOVER) wydała aż osiem megadem! W jej szeregach było kilku znakomitych koderów, takich jak METALION czy PROMAX. Zaliczali się oni do grona największych sław owych czasów. METALION, na przykład, już w 1989 roku wydał używany do dzisiaj przez wielu koderów IFF CONVERTER, PROMAX natomiast zasłynął na świecie dzięki swojemu wspaniałemu asemblerowi (SEKA 3.2). Jego konkurentem w dziedzinie asemblerów był BUDDHA z grupy SPREADPOINT, autor asemblera MasterSeka.

Wracając do KEFRENS - sam niestety nie widziałem większości z ich pierwszych megadem, jednak z tego co wiem, KEFRENS była jedną z pierwszych grup na świecie, która pokazywała w swoich demkach wypełnianą wektorówkę.

Do naszej kolekcji wspaniałych staroci należałoby również zaliczyć intra grupy SPREADPOINT, które oczywiście „kodowane" były przez BUDDHĘ. Chodzi mi o takie demka, jak na przykład MULTISCROLL DEMO, w którym na ekranie mamy tyle *scrolli*, że wręcz można dostać oczopląsu, lub też demko z lecącym „po sinusie" przestrzennym *scrollem* (takiego efektu nie wyśmianoby nawet w dzisiejszych dobrych produkcjach).

Już od 1988 roku istniał najstarszy magazyn dyskowy na świecie - niemiecki CRACKER JOURNAL. Niestety, w momencie, gdy pojawiła się konkurencja, czyli takie magazyny jak ZINE czy I.C.E., poczciwy CRACKER JOURNAL tak stracił na popularności, że z pozycji najlepszego spadł do rangi jednego z wielu przeciętnych „magów". W 1989 roku grupa CRUSADERS zaczęła wydawać będące wyrocznią na światowej scenie EURO CHARTS, czyli listy najlepszych grup, demosów, koderów, grafików, muzyków itd.



Impact Soundisk

I wreszcie w 1989 roku zaistniała na scenie nowa, rewelacyjna grupa. Mam nadzieję, że wszyscy wiedzą, kogo mam na myśli. Oczywiście RED SECTOR Inc. i ich legendarne MEGADEMO. Cóż to był za szał swego czasu na warszawskiej giełdzie. Klientów na RED SECTOR MEGADEMO było więcej, niż na najlepsze gierki. Cóż tak przyciągającego było w tym produkcie? Nie zawierał on bynajmniej nowych pomysłów. Jednak, jak już powiedziałem na początku tego artykułu, sławnymi zazwyczaj zostają nie twórcy nowego pomysłu, lecz ci, którzy potrafią go perfekcyjnie zrealizować. Niemieckie RSI MD jest tego najlepszym przykładem. Grafika stoi w tym demku na najwyższym poziomie i jest dziełem człowieka o przezwisku DARK, który niestety nie był zbyt produktywny i dziś go już niemal zapomniano. Ścieżką muzyczną zajął się ROMEO KNIGHT, który z miejsca wskoczył na „topy" muzyków we wszystkich amigowskich plebiscytach, oraz BIT ARTS, który wtedy pozostał prawie nie zauważony, natomiast ostatnio zrobił wielki „come back". Samego dema nie będę opisywał, gdyż właściwie każdy szanujący się amigowiec powinien je mieć w swoich zbiorach. Fakt jest faktem, że do roku 1990 było to najlepsze demo na Amigę.

Do tegoż roku wydano jeszcze masę innych produkcji, które jednak nie miały takiego znaczenia dla sceny, jak wspomniane wyżej. Prawdziwa eksplozja nastąpiła w roku 1990, kiedy to wydano parę przełomowych dzieł, które wpłynęły w ogromnym stopniu na styl przyszłych produkcji.

1990

W tym roku wydarzyło się naprawdę wiele. Zaczęła bowiem wyłaniać się istniejąca do dzisiaj elita i w zasadzie do końca tego roku powstały wszystkie pomysły, teraz jedynie ulepszane przez obecnie „panujących" na scenie koderów.

Ponadto w tymże roku zniknęły bezpowrotnie niektóre efekty, ponieważ demoscena zaczęła wykazywać znudzenie najpopularniejszymi do tej pory procedurami. W produktach elitarnych grup przestały pojawiać się zwykłe sinus-*scrolle*, „druciane" wektory, zwykłe *scrolle* z prostą, dużą czcionką (ang. font), nielimitowane „boby" (unlimited bobs, czyli setki tworzących określony wzór niewielkich elementów graficznych), zwykłe kolorowe paski tworzone przy pomocy Copper-a (ang. copper bars), generowany dźwięk w stylu C-64. Zniknęły też zwykłe nazwy dem (MEGADEMO xx, INTRO by xxx). Demkom, podobnie jak filmom czy teledyskom, zaczęto nadawać tytuły (na przykład MENTAL HANGOVER, HUNT FOR THE 7TH OCTOBER). Zaczęły się również pojawiać nowe pomysły, takie jak wyliczane w czasie rzeczywistym fraktale, wspaniałe, „wolne" wypełniane wektory (wolnymi wektorami nazywa się obiekty niewypukłe, czyli ze ścianami wklęsłymi), wypełniane wektorowe *scrolle* (przeważnie płaskie, a czasem nawet „grube", mające trzeci wymiar), krajobrazy przestrzenne bazowane na fraktalach. Zaczęto robić długie początki do dem (introductions), mające na celu wprowadzić oglądającego w odpowiedni nastrój, na końcu dem zaczęto umieszczać zdigitalizowane zdjęcia autorów.

I to, co najważniejsze - rok 1990 jest ostatnim rokiem, w którym wydawano MEGADEMA, czyli zlepki całkowicie oddzielnych dem, a wydano pierwsze MULTIPARTED DEMO (obecnie często nazywane „trackmem"), czyli demko stanowiące jedną całość, w którym poszczególne części przewijają się płynnie bez czynnego udziału oglądającego.

Obiecałem jednak, że nie będę sypał samymi ogólnikami i dlatego postaram się Wam przypomnieć najbardziej rewolucyjne demka, mniej więcej w kolejności chronologicznej.

CEBIT'90/RED SECTOR

To była druga rewolucja zaraz po RED SECTOR MEGADEMO. Tym produktem grupa umocniła swoją pozycję „na topie" na prawie rok. Jeżeli chociaż raz widzieliście CEBIT'90 demo, zapewne pamiętacie, że jest to demo plikowe. Ale za to jakie! Wypełnione wektory, obiekty rodem z gwiezdnych wojen, znakomity „design" (wykonanie), świetna grafika (tego samego człowieka, co w RSI MD) i muzyka ROMEO KNIGHT'a (moduł CREAM OF THE EARTH jest do tej pory uważany za jeden z najlepszych w historii Amigi). To wszystko czyni z tego file'owego (plikowego) demka prawdziwy czterdziestominutowy show. To właśnie demo spowodowało ogromny boom na wektory, bowiem prawie każda grupa marzyła o wydaniu demka z taką wektorówką. Po ukazaniu się CEBIT'90 prawie każdy amigant gustujący w demosach wymawiał z

szacunkiem imię kodera DELTA, który do tej pory pozostał legendą.

CRYPTOBURNERS MEGADEMO II

Jedno z ostatnich demek typu mega, stało jednak na o wiele wyższym poziomie, niż reszta. Grupa CRYPTOBURNERS miała bowiem w swoich szeregach takie sławy, jak HEATSEEKER (koder), dopiero debiutujący wówczas TEC (koder) oraz dwóch fantastycznych muzyków: WALKMAN i RHESUS MINUS. A na jakim poziomie stał w tym demie „koding"! Każdy z koderów pokazał swoją część z wypełnianymi wektorami, a we wszystkich z nich (w szczególności tej autorstwa TEC'a) procedury były szybsze niż w demku RSI. Niestety, „design" był nieco gorszy.

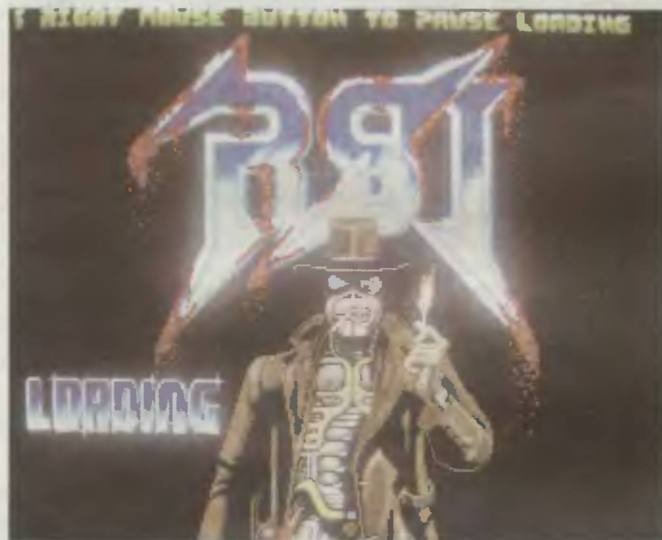
Wektorówka nie była jednak jedyną zaletą dzieła grupy CRYPTOBURNERS. Podczas wczytywania mogliśmy na przykład podziwiać jedne z pierwszych na świecie falujących „copper bar'ów". Demo to było jedną z większych rewolucji w swojej dziedzinie. Choć z pewnością nie największą, tą bowiem było...

MENTAL HANGOVER/SCOOPEX

Wniosło ono, jak do tej pory, najwięcej innowacji ze wszyskich tego typu produkcji na świecie. Nie chodzi mi wcale o nowe pomysły koderskie (choć w tym demie było ich również bardzo dużo), lecz o zupełnie nowy styl. Zacznijmy jednak od początku, gdyż ze względu na swoją rolę w historii demosceny wymaga ono dokładniejszego omówienia.

Demo zaczyna się niby normalnie, a jednak ... Na ekranie pojawia się fantastyczna ikonka z logosem (znak graficzny) SCOOPEX i dopiero po załadowaniu głównego programu uruchamia się muzyka wraz z demem, które idealnie pasuje do muzyki. I co się wtedy dowiadujemy? Z napisów na ekranie dowiadujemy się „IT ISN'T FUCKIN MEGA DEMO", co to więc jest? - „IT'S SCOOPEX DEMO".

Oczywiście sama zmiana nazwy z MEGADEMO na MENTAL HANGOVER (w dosłownym tłumaczeniu oznacza to „totalny kac") nie jest rewolucją. Główną zmianę w stosunku do normalnych MEGADEM stanowił fakt, że ogląda się je jak jedną część, przez całe demko odgrywana jest jedna muzyka, a przejście z części do części jest płynne. Oczywiście obecnie nie robi to już takiego znaczenia, gdyż prawie każde demo jest tak skonstruowane. Niewielu z Was jednak do tej pory wiedziało, że gdyby nie MENTAL HANGOVER prawdopodobnie do tej pory oglądalibyśmy same MEGADEMKA. Celowo tu oczywiście przesadzam, chcę jednak podkreślić nowatorstwo dzieła grupy SCOOPEX.



RSI Megademo

Omawiane demo jest „zakodowane" perfekcyjnie i możemy podziwiać w nim na przykład pierwszy na świecie wektorowy wypełniany *scroll* czy też „stencil vectors", czyli wektorówkę wypełnianą grafiką. Muzyka jest świetna, a grafika... Wszystkie grupy na świecie chciałyby mieć takich grafików. Autorzy tego dema natychmiast „wskoczyli" na pierwsze miejsca wszelkich rankingów demosceny (oczywiście każdy w swojej dziedzinie), napotykając jedynie na konkurencję w postaci grupy RED SECTOR.

Na zakończenie moich hymnów pochwalnych na cześć grupy SCOOPEX, przypomnę pseudonimy jej członków:

SLAYER - koder, autor i wykonawca wszystkich pomysłów,

REWARD - autor naprawdę genialnej grafiki,

UNCLE TOM - ten z kolei odpowiedzialny jest za ścieżkę muzyczną w tym demie.

BUDBRAIN MEGADEMO I

W chwili, gdy po CEBIT'90 pojawiło się MENTAL HANGOVER, wydawało się, że już nic nie będzie w stanie kogokolwiek zaskoczyć. A jednak udało się to dwóm facetom - PSYCHO i DIABLO. Ich BUDBRAIN MEGADEMO zrobiło ogromną karierę, mimo iż nie było w nim nic nowego z dziedziny kodowania, grafiki czy też muzyki. To demo było po prostu „z jajem"! Znalazło się tam pełno śmiesznych rysunków i animacji, a przy tym całość złożono bardzo dokładnie - to musiało się podobać!

KEFRENS MEGADEMO VIII

Właściwie demo to było wydane wcześniej niż reszta omawianych tu produkcji, jednak ze względu na mniejsze znaczenie opisuję je później. W demach tej grupy nigdy bowiem nie było specjalnie nowatorskich pomysłów, jednak ich dzieła oceniano bardzo wysoko, a grupa zawsze plasowała się w czołówkach światowych list przebojów. Demo to wspominam w zasadzie jako jedno z ogromnej biblioteki KEFRENS-ów - grupa ta była bowiem najproduktywniejsza na świecie do momentu rozpadu w 1991 roku. Warto przypomnieć pseudonimy trzech najsłynniejszych w 1990 roku osobistości z tej grupy:

PROMAX - koder (już w 1990 roku stworzył pierwszą wersję najlepszego asemblera na Amigę, nazwanego AsmOne),

METALLION - koder,

MAESTRO - muzyk.

HUNT FOR THE 7TH OCTOBER/CRYPTOBURNERS

Jeden z autorów CRB MEGADEMO II (patrz wyżej) jeszcze w tym samym roku powrócił na scenę z nowym produktem o bardzo oryginalnym tytule HUNT FOR THE 7TH OCTOBER. Było to demo plikowe i to w zasadzie z tylko jednym efektem, a mimo to uznano je za rewelacyjne. TEC stworzył w nim najszybszą na świecie (w owym czasie i na długo później) procedurę wypełnianych wektorów, a konkretnie: pierwszy całkiem nieźle wyglądający lot wektorowy, *scroll* wektorowy z grubych wektorowych fontów i naprawdę fantastyczne obiekty wektorowe (ponad 150) z uwzględnieniem pozycji źródła światła. Na dodatek w demku była muzyka WALKMANA, która została potem uznana przez wielu za najlepszy moduł wszechczasów - „KLIJSE PA KLIJSE".

A oto lista sławnych w 1990 roku, sporządzona według podsumowania w nieistniejącym już niestety najlepszym magazynie dyskowym świata ZINE:

NAJLEPSZE GRUPY CRACKERSKIE
1. PARADOX (rozpadła się jeszcze w tym samym roku)
2. SKID ROW
3. ANGELS (również się rozpadła)
4. DEFJAM
5. VISION FACTORY (rozpadła się, niedawno zreaktywowana)

NAJLEPSZE DEMO-GRUPY
1. SCOOPEX
2. RED SECTOR
3. CRYPTOBURNERS
4. PHENOMENA (nie wspominałem o jej produktach, gdyż w 1990 roku grupa ta tworzyła dopiero namiastkę tego, co zaprezentowała rok później)
5. KEFRENS

NAJLEPSZE MEGADEMA
1. KEFRENS MEGADEMO VIII (w tym wypadku nie zgadzam się z opinią redakcji ZINE)
2. CRYPTOBURNERS MEGADEMO II
3. BUDBRAIN MEGADEMO I

NAJLEPSZE DEMA PLIKOWE
1. MENTAL HANGOVER/SCOOPEX (ze względu na swój odmienny charakter, zaliczany był w tamtych czasach do tej kategorii)
2. CEBIT'90/RED SECTOR
3. HUNT FOR THE 7TH OCTOBER/CRB
4. FREDDY IS BACK/PARADOX
5. TRIP TO MARS/THOMAS LANDSPRUG

NAJLEPSZE DYSKI MUZYCZNE
1. DELICATE SOUNDS/RAZOR 1911
2. BACTERIA/CRUSADERS
3. SOUND OF SILENTS/THE SILENTS (praktycznie debiut KYD/BALLE PRODUCTION)

NAJLEPSI KODERZY
1. SLAYER/SCOOPEX
2. DELTA/RED SECTOR INC.
3. TEC/CRYPTOBURNERS
4. PROMAX/KEFRENS
5. FREDDY/PARADOX
6. THOMAS LANDSPURG
7. CELEBRANDIL/PHENOMENA
8. MR. GURK/PHENOMENA
9. SULKY FELLOW/VERTIGO
10. CORSAIR/SKID ROW
11. METALLION/KEFRENS

NAJLEPSI GRAFICY
1. REWARD/SCOOPEX
2. UNO/SCOOPEX
3. J.O.E./SCOOPEX
4. DARK/RED SECTOR
5. GOLEM/ANARCHY
6. MIKAEL BALLE/THE SILENTS

NAJLEPSI MUZYCY
1. UNCLE TOM/SCOOPEX
2. ROMEO KNIGHT/RED SECTOR
3. DR. AWESOME/CRUSADERS
4. 4MAT/ANARCHY
5. FIRE FOX/PHENOMENA
6. JESPER KYD/THE SILENTS
7. WALKMAN/CRYPTOBURNERS

Tak więc znamy już historię demosceny do roku 1990 włącznie. Jeżeli chcesz dowiedzieć się, co było dalej, nie masz innego wyjścia, jak tylko czekać na następny numer „C&A".

(cdn.)

NINJA/ACTION DIRECT

8520 cz. 1

COMPLEX INTERFACE ADAPTER

Układ 8520 CIA jest programowalnym portem we/wy stosowanym w Amidze. Składa się z dwóch 8-bitowych portów równoległych (PA i PB), dwóch 16-bitowych timerów (A i B), dwukierunkowego portu szeregowego (SP) i 24-bitowego licznika (event counter) z funkcją alarmu. Wszystkie te elementy mogą generować przerwania. Dostęp do 8520 jest możliwy dzięki 16 rejestrom, za pośrednictwem których dokonujemy zarówno zapisu jak i odczytu stanu portu. Procesor „widzi" te rejestry jako zwykłe komórki pamięci. Oto opis rejestrów portu 8520:

Numer	Nazwa	Spełniana funkcja
0	PRA	Rejestr danych portu A
1	PRA	Rejestr danych portu B
2	DDRA	Kierunek transmisji portu A
3	DDRB	Kierunek transmisji portu B
4	TALO	Młodsze 8 bitów timera A
5	TAHI	Starsze 8 bitów timera A
6	TBLO	Młodsze 8 bitów timera B
7	TBHI	Starsze 8 bitów timera B
8	[event low]	0-7 bity licznika
9	[event med]	8-15 bitów licznika
10	[event high]	16-23 bity licznika
11	—	Nie używany
12	SP	Rejestr danych portu szereg.
13	ICR	Rejestr kontroli przerwań
14	CRA	Rejestr kontrolny portu A
15	CRB	Rejestr kontrolny portu B

STRUKTURA PORTÓW RÓWNOLEGŁYCH

Schemat portów równoległych

Rej	Nazwa	D7	D6	D5	D4	D3	D2	D1	D0
0	PRA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
1	PRB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
2	DDRA	DPA7	DPA6	DPA5	DPA4	DPA3	DPA2	DPA1	DPA0
3	DDRB	DPB7	DPB6	DPB5	DPB4	DPB3	DPB2	DPB1	DPB0

8520 zawiera dwa 8-bitowe porty równologłe oznaczone odpowiednio PA i PB. Każdemu z nich przyporządkowany jest odpowiedni rejestr, poprzez który wysyła i odbiera się dane: PRA dla portu A i PRB dla portu B.

Układ 8520 ma 16 linii danych (PA0-PA7 i PB0-PB7). Każda linia danych może być niezależnie od pozostałych wykorzystana jako wyjście lub wejście. Każdy port dysponuje własnym rejestrem określającym kierunek transmisji - DDRA i DDRB. Zerowanie lub ustawienie bitów w tych rejestrach decyduje o kierunku przesyłania danych. Jeżeli jakiś bit w rejestrze kierunku transmisji (np. DDRA) zostanie wyzerowany, to linia danych o odpowiednim numerze zostanie ustawiona jako wejście.

Kierunek transmisji linii danych może być w każdym momencie odczytany z rejestru kierunku (DDRA lub DDRB). Przy każdym zapisie danych do portu (PRA lub PRB) wartość wysłana do tego rejestru zostaje w nim zapamiętana i znajduje sie tam dopóty, dopóki nie zostanie wpisana weń inna wartość (oczywiście tylko wtedy, gdy port ten jest ustawiony jako wyjście), a przy odczycie w rejestrze danych znajduje się stan linii wybranego portu (przy założeniu, że port jest ustawiony jako wejście).

Natomiast przy zapisie do portu ustawionego jako wejście, wartości wpisane do rejestru danych (np. DDRA) nie pojawią się na liniach danych (PA0-PA7), dopóki port nie zostanie skonfigurowany jako wyjście. I odwrotnie, jeżeli odczytasz zawartość portu ustawionego jako wyjście, to po prostu otrzymasz ostatnio wpisaną tam wartość, a nie daną z urządzenia zewnętrznego.

8520 ma dwie linie służące do przesyłania danych z potwierdzeniem (ang. handshake lines) nazwane PC i FLAG. Linia PC zmienia swój stan z wysokiego na niski na czas jednego cyklu zegara, przy każdym zapisie danej do portu B (PRB). Linia PC stanowi informację dla urządzenia zewnętrznego, że dane dla niego zostały już wpisane do rejestru B (PRB) i że może je odczytać.

Linia FLAG działa odwrotnie - informuje, że urządzenie zewnętrzne wpisało już dane do portu (lub odebrało) i że procesor może je odczytać. Dodatkowo zmiana stanu linii FLAG jest sygnalizowana poprzez ustawienie bitu „FLAG" w rejestrze kontroli przerwań (ICR) 8520.

Fakt zastosowania tych dwu linii jest znacznym ułatwieniem przy tworzeniu programów komunikujących się z urządzeniami zewnętrznymi. Program taki wysyła jedynie informację do rejestru danych portu B, a potem czeka na potwierdzenie odbioru przez urządzenie.

STRUKTURA TIMERÓW

Odczyt timera

Rej.	Nazwa	D7	D6	D5	D4	D3	D2	D1	D0
4	TALO	TAL7	TAL6	TAL5	TAL4	TAL3	TAL2	TAL1	TAL0
5	TAHI	TAH7	TAH6	TAH5	TAH4	TAH3	TAH2	TAH1	TAH0
6	TBLO	TBL7	TBL6	TBL5	TBL4	TBL3	TBL2	TBL1	TBL0
7	TBHI	TBH7	TBH6	TBH5	TBH4	TBH3	TBH2	TBH1	TBH0

Zapis do timera

Rej	Nazwa	D7	D6	D5	D4	D3	D2	D1	D0
4	PALO	PAL7	PAL6	PAL5	PAL4	PAL3	PAL2	PAL1	PAL0
5	PAHI	PAH7	PAH6	PAH5	PAH4	PAH3	PAH2	PAH1	PAH0
6	PBLO	PBL7	PBL6	PBL5	PBL4	PBL3	PBL2	PBL1	PBL0
7	PBH	PBH7	PBH6	PBH5	PBH4	PBH3	PBH2	PBH1	PBH0

8520 ma dwa 16-bitowe timery. Mogą one liczyć w „dół" od ustalonej wartości do zera. Każdy z timerów składa się z czterech rejestrów (timer A: TALO + TAHI i PALO + PAHI, a timer B: TBLO + TBHI i PBLO + BPHI).

Aby odczytać rzeczywisty stan timera, musimy najpierw zatrzymać jego pracę (port 8520 ma ośmiobitową szynę danych, więc starszy i młodszy bajt timera musimy odczytywać odzielnie). Postaram się to zilustrować przykładem:

stan timera = $0100

Z rejestru numer 5 odczytujemy starszy bajt - wartość $01. Zanim odczytamy młodszy bajt (z rejestru numer 4), timer zostanie zmniejszony - będzie wskazywał $00ff, teraz odczytamy młodszy bajt $ff, co daje fałszywą wartość $01ff.

Istnienie jednak sposób właściwego odczytania zawartości timera bez jego zatrzymywania: otóż odczytujemy starszy bajt, potem młodszy i jeszcze raz starszy. Gdy wartości starszych bajtów są równe, znaczy to, że wartość timera została odczytana prawidłowo. W przeciwnym wypadku procedurę odczytu należy powtórzyć.

Bit numer 5 rejestru kontrolnego CRA decyduje o tym, jaki sygnał będzie powodował zmniejszanie zawartości timera (A).

Dla timera A możliwe są tylko dwa źródła sygnałów.

1. Zawartość timera A będzie zmniejszana wraz z każdym taktem zegara (ponieważ w Amidze porty 8520 są podłączone do sygnału E procesora, więc częstotliwość ta wynosi 716 kHz) - jest to 1/10 częstotliwości taktowania procesora MC 68000 (INMODE = 0).
2. Zawartość timera A jest zmniejszana przez narastające zbocze sygnału CNT. (INMODE = 1).

Timer B posiada dwa bity określające źródło sygnałów, więc możliwe są ich cztery źródła:

1. i 2. - takie same jak dla timera A (bity INMODE = 00 i 01, pierwsza cyfra - bit 6, druga - 5).
3. Oba timery tworzą jeden 32-bitowy licznik (bity INMODE = 10).
4. Zawartość timera B jest zmniejszana w momencie, gdy licznik A osiągnie zero i linia CNT jest w stanie wysokim (bity INMODE = 11).

Fakt pojawienia się zera w liczniku (oznaczający koniec zliczania) jest sygnalizowany przez ustawienie odpowiednich bitów (TA i TB) w rejestrze kontroli przerwań ICR. Dla timera A jest to bit numer 0, a dla B - numer 1. Dodatkowo fakt zakończenia zliczania (pojawienie się zera) może być sygnalizowany za pomocą bitów nr 7 i 6 portu równoległego B. W tym celu należy w odpowiednich rejestrach kontrolnych (CRA i CRB) ustawić bity 'PBon'. Wtedy zakończenie zliczania impulsów będzie sygnalizowane za pomocą linii portu B (PB6 dla timera A i PB7 dla timera B).

Za pomocą bitów 'OUTMODE' możliwy jest wybór rodzaju impulsu jaki zostanie wygenerowany (bity PB6 i PB7) wraz z zakończeniem zliczania przez timery. W trybie pierwszym (bit OUTMODE = 0) koniec zliczania sygnalizowany jest dodatnim impulsem o czasie trwania jednego taktu zegara na odpowiedniej linii danych (PB6 lub PB7). Natomiast w drugim trybie (bit OUTMODE = 1), zakończenie zliczania powoduje zmianę stanu lini (z wysokiego na niski i odwrotnie).

Bit 'START' w rejestrze kontrolnym służy do rozpoczęcia (START = 1) lub zatrzymania zliczania (START = 0) osobno dla każdego timera. Timery mogą zliczać w dwu trybach - tzw. jednoimpulsowym i kontynuacji. Tryb jednoimpulsowy (RUNMODE = 1) polega na tym, że timer liczy tylko raz i potem zostaje zatrzymany (START = 0), a w trybie kontynuacji (RUNMODE = 0), po osiągnięciu zera, timer zaczyna liczyć od nowa (wartość znajduje się w wewnętrznym zatrzasku).

Zgodnie z tym, o czym wspominałem wcześniej, wpisanie wartości do rejestrów timera nie powoduje bezpośredniego zapisu do rejestru liczącego, lecz tylko do wewnętrznego zatrzasku (zwanego także prescaler, ponieważ liczba timeouts w ciągu sekundy jest równa częstotliwości zegara podzielonej przez wartość wpisaną do prescaler). Istnieją sposoby załadowania wartości z zatrzasku do rejestru liczącego:

1. Ustawić bit 'LOAD' w rejestrze kontrolnym, co powoduje natychmiastowe wpisanie wartości z zatrzasku bez względu na stan timera. Bit 'LOAD' jest inaczej zwany bitem strobu.

2. Drugi sposób jest bardzej „naturalny": kiedy timer rozpoczyna cykl liczenia, wartość z zatrzasku jest automatycznie wpisywana do timera.

3. Podczas wpisywania liczby do rejestru, w którym znajduje się starszy bajt wartości timera (rejestr 5 dla A i 7 dla B) bit 'START' zostaje wyzerowany i wartość z zatrzasku jest wpisana do timera, po czym 'START' jest ustawiany i timer zaczyna liczyć (wpisanie młodszego bajtu nie powoduje tego, więc należy go zainicjować najpierw).

(cdn.)

JERZY DUDEK
KRZYSZTOF BORKOWSKI

W następnym miesiącu dowiecie się jak działa zegar czasu rzeczywistego TOD i rejestr kontroli przerwań oraz jak przebiega transmisja poprzez port szeregowy.

Znaczenie bitów w rejestrze kontrolnym A (Rejestr nr 14 - CRA)

Numer bitu	Nazwa	Stan/funkcja
0	START	1 - start, 0 - stop timera
1	PBon	czy [timeouts] sygnalizowane w PB - 1=tak
2	OUTMODE	1 - impuls, 0 - zmiana stanu lini
3	RUNMODE	1 - jednoimpulsowy, 0 - kontynuacji
4	LOAD	dodatni impuls (1) - wpisanie zatrzasku
5	INMODE	0 - zegar, 1 - CNT
6	SPMODE	0 - wejście, 1 - wyjście (port szeregowy)
7	-	nie używany

Znaczenie bitów w rejestrze kontrolnym portu B (Rejestr nr 15 - CRB)

Numer bitu	Nazwa	Stan/funkcja
D6 + D5	INMODE	00 - zegar 01 - CNT 10 - timer A 11 - timer A + CNT
D7	ALARM	1 - TOD, 0 - alarm

Bity od zerowego do czwartego mają identyczną funkcję co w porcie A.

The event counter

Rej.	Nazwa	D7	D6	D5	D4	D3	D2	D1	D0
8	LSB event	E7	E6	E5	E4	E3	E2	E1	E0
9	event 6-15	E15	E14	E13	E12	E11	E10	E9	E8
10	MSB event	E23	E22	E21	E20	E19	E18	E17	E16

Timer

Specjalny rejestr, który automatycznie zwiększa lub zmniejsza swoją zawartość. Może służyć wielu celom, np. generowaniu liczb losowych, jednak jego głównym przeznaczeniem jest odmierzanie czasu - stąd nazwa (po ang. time: czas).

W portach 8520 CIA timery zmniejszają swą zawartość. Gdy do któregoś z rejestrów timera wpisujemy liczbę, zostaje ona najpierw zapamiętana w tzw. zatrzasku, skąd następnie zostaje wpisana do rejestru liczącego, gdzie jest zmniejszana aż do zera; wtedy liczba zapamiętana w zatrzasku zostaje z powrotem wpisana do tego rejestru i cały proces liczenia się powtarza.

O PRZERWANIACH SŁÓW KILKA

Przerwania procesora MOTOROLA inaczej nazywane są stanami wyjątkowymi. Powinniśmy zadać sobie jednak na początku pytanie: co to jest przerwanie lub też stan wyjątkowy procesora? Mówiąc najprościej, jest to sytuacja, kiedy wykonywanie zwykłego programu zostaje przerwane i procesor „skacze" do specjalnego programu obsługującego przerwanie. Oczywiście, po wykonaniu programu obsługi przerwania następuje powrót do głównego zadania.

Ktoś mógłby zapytać, po co używa się przerwań? Odpowiedź jest prosta: wyobraźmy sobie, że mamy procedurę, którą musimy wielokrotnie wykonywać w regularnych odstępach czasu. Normalnie byłoby to bardzo skomplikowane, natomiast przerwania wykonują to za nas. Jako inny przykład mógłbym podać sposób komunikowania się komputera z urządzeniami zewnętrznymi, które są zazwyczaj wolniejsze od naszego procesora. Zastosowanie przerwań pozwala na zwiększenie szybkości wykonywania programów, procesor nie musi czekać, aż urządzenie zewnętrzne będzie gotowe do przyjęcia lub wysłania informacji, może swobodnie wykonywać swój program. Jeżeli urządzenie będzie już gotowe, wymusi przerwanie na procesorze, który przerwie pracę i zajmie się urządzeniem.

Procesor MOTOROLA 68000 ma 256 wektorów stanów wyjątkowych. Wektory są adresami programów obsługujących określone przerwania, z których to korzysta procesor podczas wywoływania przerwania. Wektorów jest 256, ale my zajmiemy się tylko siedmioma, oznaczonych numerami od 25 do 31.

Wektor	Adres	Poziom
25	$64	Autowektor (poziom 1)
26	$68	Autowektor (poziom 2)
27	$6C	Autowektor (poziom 3)
28	$70	Autowektor (poziom 4)
29	$74	Autowektor (poziom 5)
30	$78	Autowektor (poziom 6)
31	$7C	Autowektor (poziom 7) niedostępny

Są to tzw. autowektory, które umieszczono od adresu $000064 do $00007C. Każdy z autowektorów obsługuje inny priorytet przerwania. Należy pamiętać, że przerwanie, które ma wyższy priorytet, ma pierwszeństwo nad przerwaniem o niższym. Przerwanie o siódmym poziomie ważności (priorytet 7) jest przerwaniem niemaskowalnym, niedostępnym dla programisty.

Amiga posiada specjalny układ zarządzający poszczególnymi źródłami przerwań i generujący sygnały przerwań dla Motoroli - zajmuje się tym obwód w kości specjalizowanej Paula. Do dyspozycji mamy cztery rejestry:

- rejestry maski przerwania:
  INTENA $DFF09A (do zapisu)
  INTENAR $DFF01C (do odczytu)

- rejestry żądania przerwania:
  INTREQ $DFF09C (do zapisu)
  INTREQR $DFF01E (do odczytu)

Przeznaczenie bitów w obu typach rejestrów jest takie samo:

Ustawianie i zerowanie bitów w obu rejestrach (INTENA i INTREQ) odbywa się w taki sam sposób, jak w przypadku rejestru DMACON. Jeżeli wpisujemy słowo danych, o tym, czy bity rejestru zostaną ustawione, czy też wyzerowane, decyduje bit 15. Jeżeli we wpisywanym słowie bit ten będzie ustawiony, to wszystkie bity mające w tym słowie stan logiczny 1 zostaną ustawione w rejestrze. I na odwrót - jeżeli bit 15 będzie wyzerowany, to wszystkie ustawione bity słowa zostaną w rejestrze wyzerowane. Jeżeli chcemy ustawić np. bit 4 w rejestrze INTENA musimy wpisać:

MOVE #%1000000000010000,$DFF09A

Natomiest jeżeli chcemy ten bit wyzerować, wpiszemy:

MOVE #%0000000000010000,$DFF09A

W rejestrze INTENA znajduje się bit INTEN (bit 14), który pełni funkcję głównego przełącznika. Oznacza to, że przerwania mogą być generowane dopiero, gdy bit ten zostanie ustawiony. Natomiast wyzerowanie tego bitu spowodu-

Numer bitu	Nazwa	Poziom	Objaśnienie
15	SET/CLR		
14	INTEN	(6)	Główny przełącznik
13	EXTER	6	Przerwanie z CIA-B lub portu rozszerzeń
12	DSKSYN	5	Rozpoznanie wartości synchronizacji dysku
11	RBF	5 s	Pełen bufor odbiorczy danych zeregowych
10	AUD3	4	Wyjście danych dźwiękowych kanał 3
9	AUD2	4	Wyjście danych dźwiękowych kanał 2
8	AUD1	4	Wyjście danych dźwiękowych kanał 1
7	AUD0	4	Wyjście danych dźwiękowych kanał 0
6	BLIT	3	Blitter gotowy
5	VERB	3	Początek wygaszania pionowego
4	COPER	3	Przerwanie Coppera
3	PORTS	2 rozszerzeń	Przerwanie z CIA-A lub portu
2	SOFT	1	Przerwanie programowe
1	DSKBLK	1	Transfer przez DMA dysku
0	TBE	1 szeregowych	Pusty bufor nadawczy danych

je zablokowanie wszystkich przerwań, a tym samym praca naszego programu nie będzie przez nic zakłócana.

W rejestrze INTREQ bit 14 ma troszeczkę inne przeznaczenie, jego ustawienie spowoduje wygenerowanie przerwania o poziomie 6 (adres procedury obsługującej przerwanie znajduje się pod adresem $78).

Przerwanie jest generowane dopiero w momencie, kiedy zostaną spełnione odpowiednie warunki:
- bit INTEN w rejestrze INTENA będzie ustawiony,
- bit odpowiedniego przerwania w rejestrze INTENA zostanie ustawiony,
- bit odpowiedniego przerwania (ten sam co w INTENA) w rejestrze INTREQ zostanie ustawiony (w tym momencie jest generowane przerwanie).

Procedura obsługująca przerwanie powinna mieć odpowiedni schemat, który gwarantuje, że komputer się nie zawiesi:
- na początku odkładamy na stos wartości rejestrów używanych przez procedurę obsługi przerwania,
- dalej wykonywana jest nasza procedura,
- na końcu pobieramy ze stosu wartości i przywracamy rejestrom stan sprzed przerwania,
- przed powrotem do głównego programu musimy wyzerować bit odpowiedniego przerwania w rejestrze INTREQ (będzie to informacja dla komputera, że przerwanie już wystąpiło),
- każdą procedurę obsługi przerwania należy zakończyć rozkazem RTE (pamiętaj - nie RTS).

Z boku przedstawiam przykładowy program, który wykorzystuje przerwanie autowektorowe o trzecim poziomie ważności (ma trzeci priorytet). Jest to przerwanie wygaszania pionowego obrazu, jest więc generowane na początku każdej ramki ekranu, co 1/50 sekundy. Po ustawieniu adresu procedury obsługi przerwania, ustawieniu bitu INTEN (bit 14) i bitu VERB (bit 5) w rejestrze INTENA, przerwanie będzie generowane 50 razy na sekundę. Bit VERB w rejestrze INTREQ będzie ustawiany automatycznie, my musimy jedynie pamiętać o jego wyzerowniu po wykonaniu procedury.

Procedura zajmuje się odczytem ruchu myszki: w komórce oznaczonej etykietą „PozXY" ustawia odpowiednie wartości, na podstawie których program główny może określić, w którą stronę została przesunięta myszka. Każdemu ruchowi towarzyszy zmiana koloru tła. Jeżeli chcesz wyjść z programu, wciśnij lewy przycisk myszki.

BARTOSZ SMAGA

```
        ;PROGRAM 1
        ;Odczyt ruchu myszki

Start:  lea     STAREINTENA(pc),a0
        move.w  $dff01c,(a0)
        lea     WEKTORVERTB(pc),a0
        move.l  $6c,(a0)
        move.w  #$4070,$dff09a
        move.w  #$0070,$dff09c
        lea     READMOUSE(pc),a0
        move.l  a0,$6c          ;wpisanie adresu przerwania
        move.w  #$c020,$dff09a  ;włączenie przerwania

KlawiszMyszy:   bsr OdczytRuchu
        btst    #6,$bfe001
        bne.b   KlawiszMyszy
        move.w  #$4000,$dff09a
        lea     WEKTORVERTB(pc),a0
        move.l  (a0),$6c
        lea     STAREINTENA(pc),a0
        move.w  (a0),d0 or.w #$C000,d0
        move.w d0,$dff09a
        rts

STAREINTENA:    dc.w    0
WEKTORVERTB:    dc.l    0
 ;Podprogram odczytuje wartości z komórki określonej
 ;etykietą PozXY i na jej podstawie uruchamia odpowiednie
 ;podprogramy: MyszkaDoDolu, MyszkaDoGory, MyszkaWLewo,
 ;MyszkaWPrawo.

OdczytRuchu:
        lea     PozXY(pc),a0
    ;załadowanie do D0 kodów ruchu myszy
        move.l  (a0),d0
    ;sprawdzenie niższych 15 bitów
        cmpi.w  #-1,d0
        bne.b   OdczRuchp1
        bsr.w   MyszkaDoDolu
        bra.b   OdczRuchp2

OdczRuchp1:  cmpi.w  #1,d0
        bne.b   OdczRuchp2
        bsr.w   MyszkaDoGory

OdczRuchp2:  swap  d0 ;bity od 16 do 31 zostają podstawione
                   ;pod bity 0-15 (i na odwrót)
        cmpi.w  #-1,d0
        bne.b   OdczRuchp3
        bsr     MyszkaWLewo
        bra.w   OdczRuchRts

OdczRuchp3:  cmp.w   #1,d0
        bne.w   OdczRuchRts
        bsr.b   MyszkaWPrawo

OdczRuchRts:
        rts

MyszkaDoDolu:   ;podprogram zostanie wykonany, kiedy
                ;przesuniesz myszkę do dołu

        move.w  #$100,$dff180   ;zmiana koloru tła na
        rts                     ;czerwony

MyszkaDoGory:   ;podprogram zostanie wykonany, kiedy
                ;przesuniesz myszkę do góry

        move.w  #$010,$dff180   ;zmiana koloru tła na
        rts                     ;niebieski

MyszkaWLewo:    ;podprogram zostanie wykonany jeżeli
                ;przesuniesz myszkę w lewo
        move.w  #$001,$dff180   ;kolor tła będzie zielony
        rts

MyszkaWPrawo:   ;podprogram zostanie wykonany jeżeli
                ;przesuniesz myszkę w prawo
        move.w  #$fff,$dff180   ;kolor tła będzie biały
        rts

    ;PROGRAM 2
    ;Procedura odczytu ruchu myszki
    ;(pracuje na autowektorze 3)

READMOUSE:
        movem.l d0/d1/d2/d3/a0,-(sp)
        clr.l   d0
        clr.l   d1
        clr.l   d2
        clr.l   d3
        move.b  $dff00a,d0      ;licznikY-nowy
        move.b  $dff00b,d1      ;licznikX-nowy
        lea     StaraPozMouse(pc),a0
        move.b  (a0),d2         ;licznikY-stary
        move.b  1(a0),d3        ;licznikX-stary
        move.b  d0,(a0)         ;zapamiętanie nowego Y
        move.b  d1,1(a0)        ;zapamiętanie nowego X
        lea     PozXY(pc),a0
        subi.b  d0,d2           ;w pionie
        bpl.b   RDMp1           ;skok jeśli nie w dół
        move.w  #-1,2(a0)
        bra.b   RDMp2

RDMp1:  bne.b   RDMp21          ;skok jeśli nie w górę
        move.w  #1,2(a0)
        bra.b   RDMp2

RDMp21: ;nie było ruchu w pionie
        move.w  #0,2(a0)

RDMp2:  subi.b  d1,d3
        bpl.b   RDMp3           ;skok jeśli nie w prawo
        move.w  #1,(a0)
        bra.b   RDMrte

RDMp3:  beq.b   RDMp4           ;skok jeśli nie w lewo
        move.w  #-1,(a0)
        bra.b   RDMrte

RDMp4:  ;nie było ruchu w poziomie
        move.w  #0,(a0)

RDMrte:
        movem.l (sp)+,d0/d1/d2/d3/a0
        move.w  #$0020,$dff09c     ; zerowanie bitu VERTB
w
                                   ; rejestrze INTREQ
        rte

StaraPozMouse:  dc.w    0
PozXY:          dc.w    0,0
```

AMOS (cz. 5)

PROCEDURY - c.d.

W tym odcinku dokończymy kwestię procedur, krótko omówimy operacje na łańcuchach i dodamy coś nowego do tego, co wiemy już o grafice.

Aby skompletować wszystkie wiadomości dotyczące procedur, powinniśmy jeszcze poznać sposób wychodzenia z nich w innym miejscu, niż na końcu. Wyobraźmy sobie krótki programik:

```
Global KOD1$
KOD1$="c&a"
Input "Podaj kod dostepu:  >>";KOD2$
PROGRAM[KOD2$]
End
'
Procedure PROGRAM[K$]
   If K$<>KOD1$
       Print "KOD NIEPRAWIDLOWY"
         Print "Niestety nie masz dostepu do wnetrza programu."
       Goto KONIEC
   End If
   Print "KOD PRAWIDLOWY"
     Print "Masz teraz dostep do wszytkich danych."
   Print "Oto one:"
   Print
   Print "C&A - Commodore & Amiga"
   Print "Warszawa"
   Print "ul. Wasilkowskiego 7"
   Print "Tel. 643-18-40"
   KONIEC:
End Proc
```

Program sprawdza, czy podane przez użytkownika hasło jest takie samo, jak w zmiennej KOD1$. Jeżeli porównanie wypadło negatywnie, zostaje wykonany skok na koniec procedury z pominięciem instrukcji znajdujących się po wyrażeniu warunkowym "If..End If". Lepiej byłoby, gdyby wyjście z procedury następowało zaraz po komunikacie o niepowodzeniu. Zaoszczędzilibyśmy sobie tworzenia zbędnej etykiety, a program stałby się krótszy.

Wstawmy więc w miejsce instrukcji "Goto KONIEC" nową: "Pop Proc", a także wykasujmy niepotrzebną już etykietę "KONIEC:". Czy wszystko teraz jest w porządku? Myślę, że nie trzeba tego dokładniej tłumaczyć.

OPERACJE NA ŁAŃCUCHACH

Jest to ostatni zestaw instrukcji podstawowych, po którego poznaniu będziemy mogli przejść do rzeczy nieco bardziej skomplikowanych.

Jak chyba wiadomo, dobry program powinien się w miarę możliwości dość dokładnie komunikować z użytkownikiem. Komunikacja ta powinna zachodzić w obie strony. Nie zawsze także odbywa się ona tylko za pomocą liczb - często za pomocą słów, czyli tekstu. O ile człowiekowi jest dość łatwo zinterpretować wyświetlone na ekranie informacje (rysunek, wykres, liczby, komunikaty), o tyle w przypadku komputera sprawa jest nieco trudniejsza.

Komputer rozpoznaje jedynie ściśle określone dane. Weźmy na przykład program, który napisaliśmy przy omawianiu procedur. Wystarczy zamiast "c&a" wpisać "C&A" i już hasło nie zostanie rozpoznane, choć z punktu widzenia człowieka jest to to samo. Dobry programista musi pisać programy, które nie powinny się "dać zaskoczyć" w taki sposób. Do tego, a także do innych celów, mogą służyć podane poniżej instrukcje.

Left$(ZMIENNA$,N)

Instrukcja ta powoduje wybranie z łańcucha tekstowego pierwszych N znaków licząc od lewej strony. Można w ten sposób np. skrócić podany przez użytkownika tekst, który okazuje się być za długim. Oto przykład:

```
Input "Podaj 5-znakowe haslo:";HASLO$
HASLO$=Left$(HASLO$,5)
Print "Haslo, ktore wprowadziles to:"
Print HASLO$
```

Jeżeli teraz wprowadzimy hasło dłuższe niż pięć znaków, to te niepotrzebne zostaną jakby odcięte. Spróbujcie sami!

Right$(ZMIENNA$,N)

Instrukcja podobna w działaniu do poprzedniej - wydziela odpowiednią liczbę (N) znaków począwszy od prawej strony łańcucha tekstowego. Przykład pomijam, gdyż używa się jej tak samo jak instrukcji poprzedniej.

Mid$(ZMIENNA$,N,ILE)

Instrukcja ta powoduje wybranie ILE znaków począwszy od N-tego miejsca licząc od lewej strony. W praktyce oznacza to, że możemy sobie wydzielić jakąś ilość znaków ze środka łańcucha tekstowego. Nim podam przykład poznajmy jeszcze jedną, prostą instrukcję:

Len(ZMIENNA$)

Powoduje ona podanie liczby całkowitej będącej długością łańcucha tekstowego ZMIENNA$. A teraz przykład:

```
TXT$="AMOS to wspanialy jezyk programowania"
For N=1 To Len(TXT$)
   Print Mid$(TXT$,N,1);
   Wait 25
Next N
```

Może to być jeden z najprostszych sposobów urozmaicenia działania naszych programów. Instrukcja Wait, której jeszcze nie omawialiśmy, powoduje wstrzymanie działania programu na czas podany w parametrze, liczony w pięćdziesiątych częściach sekundy.

Instrukcje "Left$()", "Right$()" i "Mid$()" mogą być używane także jako funkcje, tzn. za ich pomocą można zmieniać wybrany fragment łańcucha tekstowego, np.:

Left$(ZM$,4)="Amos" - 4 pierwsze znaki łańcucha ZM$ (licząc od lewej) zostaną zamienione na ciąg "Amos";

Right$(ZM$,5)="Amiga" - 5 ostatnich znaków zmiennej tekstowej ZM$ zostanie zmienione na łańcuch "Amiga";

Mid$(ZM$,4,3)="C&A" - znaki od 4 do 7 włącznie zmiennej ZM$ zostaną zmienione na "C&A".

A oto przykład praktycznego zastosowania jednej z tych instrukcji:

```
ZM$="Kazdy wie, ze Atari to swietny komputer!"
Mid$(ZM$,15,5)="Amiga"
Print ZM$
```

A teraz następna instrukcja:

```
Instr(ZM1$,ZM2$)
```

Powoduje ona przeszukanie łańcucha znaków ZM1$ w celu znalezienia łańcucha ZM2$. Jeżeli operacja ta zakończy się sukcesem, to w wyniku otrzymamy pozycję pierwszego znaku z odszukanego ciągu. Jeżeli łańcuch znaków ZM2$ nie występuje w ZM1$, to w wyniku otrzymamy zero. Instrukcja ta może mieć także bardziej rozbudowany format:

```
Instr(ZM1$,ZM2$,ODKAD)
```

Liczba ODKAD wskazuje, od którego znaku zmiennej ZM1$ ma się rozpocząć czytanie. A teraz przykład:

```
Input "Podaj ciag do przeszukania: ";CIAG1$
Input "Podaj ciag do szukania: ";CIAG2$
Input "Od ktorego znaku mam szukac? ";POCZ
POZYCJA=Instr(CIAG1$,CIAG2$,POCZ)
If POZYCJA=0
```

```
  Print "Niestety ciag: '";CIAG2$;"' nie wystapil w ciagu: '";CIAG1$
Else
  Print "Ciag: '";CIAG2$;"' wystepuje w ciagu: '";CIAG1$;
  Print "' poczawszy od";POZYCJA;" znaku."
End If
```

Lower$(ZM$)
Instrukcja ta przyjmuje wartość łańcucha znaków ZM$, z tym tylko, że wszystkie duże litery zostaną zamienione na małe. Jako przykład zastosowania proponuję usprawnić pierwszy program, który dziś napisaliśmy. Jak zapewne zauważyliście, podanie zamiast „c&a" hasła np. „C&A" spowoduje jego nierozpoznanie. Wstawmy więc tuż pod linią, w której występuje instrukcja „Input", następujący wiersz:

```
KOD2$=Lower$(KOD2$)
```

Od tej pory wielkość wpisywanych jako hasło liter nie będzie miała znaczenia, gdyż komputer i tak zamieni je sobie na małe.

Upper$(ZM$)
Instrukcja działa analogicznie do poprzedniej, ale wszystkie znaki zamieniane są na duże litery.

Flip$(ZM$)
Instrukcja ta przyjmuje wartość ciągu składającego się ze znaków zmiennej ZM$, ale w odwróconej kolejności. Krótko mówiąc, zmienia tekst tak, że czytać go możemy potem „od tyłu". Oto przykład, który zawiera jednak pewien trik. Poczytajcie:

```
ZM$="KOBYLA MA MALY BOK"
ZM$=Flip$(ZM$)
Print ZM$
```

Space$(N)
Instrukcja ta przyjmuje wartość ciągu znaków złożonego z N spacji (znaków odstępu).

String$(ZM$,N)
Przyjmuje wartość łańcucha złożonego z pierwszego znaku zmiennej ZM$. Jego długość ustala liczba N, np.:

```
Print String$(„Amiga",10)
```

da na ekranie wydruk:
AAAAAAAAAA

Chr$(N)
Instrukcja ta tworzy jednoznakowy łańcuch, o kodzie ASCII równemu N. Jako przykład podam teraz krótki programik drukujący na ekranie tabelę kodów ASCII. Proszę nie sugerować się instrukcjami warunkowymi - służą one jedynie do formatowania wydruku, ponieważ nie znamy chwilowo bardziej zaawansowanych komend.

```
Screen Open 0,656,256,8,Hires
Print „TABELA KODOW ASCII"
Print String$(„-",78)
For N=32 To 255
  If N>100
    If(N-4) mod 7=0
      Print
    End If
  Else
    If(N-4) mod 9=0
      Print
    End If
  End If
  Print N;" - „;Chr$(N),
Next N
```

Asc(ZM$)
Instrukcja ta tworzy zmienną całkowitą równą wartości kodu ASCII pierwszego znaku zmiennej ZM$. Jest to jakby przeciwieństwo poprzedniej instrukcji. Przykładem zastosowania niech będzie taki program:

```
Input „Wprowadz ciag znakow: „;CIAG$
Print „Jego kolejne kody ASCII to:"
For N=1 To Len(CIAG$)
  Print Asc(Mid$(CIAG$,N,1))
Next N
```

Val(ZM$)
Jeżeli zmienna tekstowa jest postaci „3.14159" to czasami może zależeć nam na tym, aby komputer zamiast ciągu znaków przyjął wartość liczby za pomocą tych znaków napisanej. Krótko mówiąc instrukcja ta powoduje, że komputer próbuje zamienić tekst na liczbę, jeżeli mu się to jednak nie uda, to wstawia wartość zero, np.:

```
A$="Amiga"
A=Val(A$)
Print A
```

(wynikiem tej operacji będzie 0), albo:

```
A$="1992"
A=Val(A$)
Print A
(wynik: 1992)
```

Str$(N)
Ta instrukcja działa odwrotnie niż poprzednia, tzn. zamienia tekst na liczbę, np.:

```
A=15673
A$=Str$(a)
Print A$
(wynik: 15673)
```

GRAFIKA

W poprzednim odcinku poznaliśmy sposób otwierania tzw. screenów, czyli nieodłącznych dla grafiki elementów (są one miejscem, gdzie możemy coś rysować, pisać itp.). Dziś kilka podstawowych instrukcji służących do umieszczania na ekranie czegoś innego, niż tylko sam tekst.

Plot(X,Y)
Powoduje wykreślenie na ekranie punktu o współrzędnych X,Y. Jak już niegdyś wspomniałem, ekran podzielony jest na punkty, a każdy z nich ma dwie współrzędne: poziomą (X) i pionową (Y). Wszystko to przypomina prostokątny układ współrzędnych żywcem wyjęty ze szkolnego zeszytu. Jedyną różnicą jest fakt umieszczenia początku tego układu w lewym górnym rogu screenu, a oś Y przyrasta w dół. Maksymalne współrzędne, jakie można podać, zależą od trybu rozdzielczości danego screenu (patrz: numer poprzedni, otwieranie screenów). Podanie współrzędnych nie mieszczących się w danym screenie nie powoduje błędu, lecz punkt będzie na ekranie niewidoczny.

Draw X1,Y1 to X2,Y2
Rysuje linię prostą zawartą między współrzędnymi X1,Y1, a X2,Y2. Można tu pominąć pierwsze dwie współrzędne - linia rysowana będzie wtedy od ostatniego poprzednio namalowanego punktu. Jeżeli takiego punktu nie ma, to linia rysowana będzie od początku ekranu (tzn. od punktu 0,0). Np.:

```
Draw 30,50 To 120,150
```

lub też:

```
Draw To 200,75
```

Polyline X1,Y1 To X2,Y2 To X3,Y3 (...)
Instrukcja powoduje wykreślenie na ekranie linii łamanej złożonej z dowolnej ilości linii zawartych między współrzędnymi odpowiednio X1,Y1 i X2,Y2; X2,Y2 i X3,Y3 itd. Np.:

```
Polyline 50,50 To 100,100 To 150,50 To 200,150
```

Box X1,Y1 To X2,Y2
Używanie tej instrukcji jest podobne do „Draw ... To ...", tylko że między współrzędnymi rysowany jest niewypełniony prostokąt. Np.:

```
Box 10,10 To 310,190
```

Circle X,Y,R
Instrukcja to powoduje narysowanie okręgu o środku w punkcie X,Y i o promieniu R, np.:

```
Circle 160,100,50
```

I to już wszystko w tym odcinku kursu. Obiecuję, że stopniowo coraz więcej uwagi będę poświęcał grafice i dźwiękowi. Do tej pory tak nie było, ponieważ założyłem sobie, że najpierw zapoznam z podstawami programowania tych, którzy stykają się z tym problemem pierwszy raz. Aby stosować różne efektowne sztuczki, trzeba niestety poznać dokładnie podstawy danego języka.
To tyle dla zniechęconych i namawiam do wszelkich eksperymentów, jakie tylko może podsunąć Wam, drodzy Czytelnicy, Wasza wyobraźnia. Do zobaczenia za miesiąc.

RAFAŁ BORZYŃSKI (RABOCOST)

SŁOWNICZEK

SCREEN - dosłownie: ekran. Doprowadza to jednak do paradoksu, ponieważ na jednym (fizycznym) ekranie monitora może być nawet kilkanaście screenów. Wobec takiego stanu rzeczy używam oryginalnej pisowni tego wyrazu.
Screen to jeden z podstawowych elementów środowiska graficznego Amigi. Na nim dopiero mogą być otwierane okna, w których wszystko w tym komputerze się "odbywa".
ASCII - znormalizowany dla niemal wszystkich komputerów system kodów, w jakim zapisywane są znaki (komputer każdy znak tekstu - literę lub cyfrę - zapamiętuje w postaci liczby).

ASEMBLER 68000 (cz.V)

EKRAN AMIGI

W Polsce jest bardzo wielu amigowców, ale niewielu z nich wie, w jaki sposób ich komputer tworzy na ekranie monitora obraz. Początkujący amigowcy nie zdają sobie do końca sprawy, w jakich trybach graficznych pracuje ich komputer: jakie są rozdzielczości ekranu, ile można wyświetlić na ekranie kolorów. Zdolności graficzne Amigi są nadal dla wielu osób owiane tajemnicą.

Rozdzielczości

W Amidze wyświetlaniem obrazu zajmuje się specjalizowany układ DENISE, który po otrzymaniu od układu AGNUS danych graficznych z pamięci RAM przekształca je na odpowiednie odwzorowania RGB i wyprowadza na ekran. Nasz komputer ma kilka trybów rozdzielczości.

Ktoś mógłby w tym miejscu zapytać: ale co to jest rozdzielczość? Obraz złożony jest z pojedyńczych elementów nazywanych pikselami (ang. pixel - punkt). Cały ekran podzielony jest na linie (jest to poziomy rząd pikseli) i kolumny (kolumna ma szerokość jednego piksela). Rozdzielczością nazywamy stosunek liczby kolumn do liczby linii. Od rozdzielczości ekranu zależy dokładność wyświetlanych obrazów. Dla dużych rozdzielczości pojedyńczy element obrazu jest bardzo mały i rysunki mogą być dokładniejsze, bardziej skomplikowane, ale będą zajmowały więcej pamięci.

W starszych Amigach instalowano kości DENISE oznaczone numerem 8362. Jeżeli Amiga wyposażona jest w tą kość, to ma możliwość wyświetlania dwóch rozdzielczości poziomych i dwóch rozdzielczości pionowych. W Amigach 500+, 600 i 3000 zainstalowano nową kość - DENISE 8373. Dzięki nowemu układowi Amiga oferuje dodatkową rozdzielczość poziomą. Poniżej znajduje się tabela, w której zebrano dane na temat dwóch kości DENISE - jakie pozwalają uzyskać rozdzielczości i ile mamy do dyspozycji kolorów przy każdej z nich.

Budowa obrazu

Na ilustracji nr 1 pokazano standardowy układ ekranu w Europie. Dlaczego w Europie? Amigi produkowane są w dwóch różnych wersjach, dla dwóch różnych systemów (PAL i NTSC), i w zależności od rodzaju wersji różnią się kością specjalizowaną AGNUS: amerykańska wersja - NTSC i europejska - PAL. W systemie PAL obraz składa się z 625-ciu linii, gdy w przypadku NTSC - tylko z 525-ciu.

Tabela 1

Rozdzielczości	320x256 320x256	640x256 640x512	1240x256 1240x512
Liczba kolorów (DENISE 8362)	2,4,8,16, 32,64,4096	2,4,8,16	-
Liczba kolorów (DENISE 8373)	2,4,8,16, 32,64,4096	2,4,8,16	2,4

Jak na ekranie tworzony jest obraz? Jak już wiemy, obraz składa się z linii, każda z linii złożona jest z pikseli (pojedyńczych punktów ekranu), które zapalane są przez wiązkę elektronów biegnących od lewej do prawej strony każdej linii. Po każdej linii następuje krótka przerwa (tzw. Horizontal Blanking Gap - przerwa wygaszania poziomego), podczas której wiązka elektronów przesuwana jest na lewą stronę, na początek następnej linii.

Wiązka elektronów przebiega ekran od górnego lewego rogu do dolnego prawego, potem przesuwana jest na powrót do górnego lewo rogu i cykl się powtarza. Obraz, aby był widoczny na ekranie, musi być wciąż odświeżany (wiązka elektronów musi wciąż zapalać odpowiednie piksele) i w systemie PAL odbywa się to 50 razy na sekundę (w NTSC - 60 razy). Przerwa, podczas której wiązka przechodzi z prawego dolnego rogu ekranu do lewego górnego (ang. Vertical Blanking Gap - przerwa wygaszania pionowego), jest bardzo często wykorzystywana przez programistów (podczas VBG wywoływane jest przerwanie poziomu 3).

Łatwo obliczyć, że jeżeli wszystkie 625 linii miałyby być rysowane na ekranie 50 razy w ciągu sekundy, to łączna liczba wszystkich linii rysowanych na ekranie wyniosłaby 31250 na sekundę. Sprzęt o takich parametrach był, w czasie, gdy konstruowano Amigę, bardzo drogi, dlatego też zastosowano pewną sztuczkę. Linie podzielono na dwie grupy: parzyste (2,4,6,...624) i nieparzyste (1,3,5,...625). Dwa takie półobrazy (tzw. ramki) tworzą cały obraz i są wyświetlane na ekranie na przemian: raz linie nieparzyste (tzw. Long Frame - długa ramka), potem linie parzyste (Short Frame - krótka ramka). Linii nieparzystych jest więcej (313) niż linii parzystych (312), dlatego też ich wyświetlenie trwa troszeczkę dłużej, i stąd też wzięła się nazwa ramek.

Przy tej technice liczba kompletnych obrazów na sekundę spadła do 25, nie ma więc problemu z jednoczesnym wyświetlaniem 625 linii na ekranie. Pojawił się za to inny problem - migotanie. Migotanie pojawia się w obrazach wysokiej rozdzielczości, gdy kontur ograniczony jest tylko do jednej linii (w jednej ramce jest on rysowany, w drugiej go nie ma). Problem ten można częściowo wyeliminować odpowiednio dobierając kolory i unikając gwałtownych przejść między płaszczyznami bardzo jasnymi i ciemnymi. W obrazach niskiej rozdzielczości obraz nie miga, ponieważ obie ramki są identyczne i ich częstotliwość wynosi 50 Hz (50 ramek na sekundę). Liczba linii jest wtedy ograniczona do 313.

Obraz wyświetlany na ekranie jest reprezentowany w pamięci za pomocą tzw. płaszczyzny bitowej (ang. bitplane; w dalszej części artykułu będę używał "polskiej wersji" tego słowa - bitplan). Jest to nic innego, jak ciągły blok pamięci, w którym ustawienie bitu spowoduje zapalenie się odpowiedniego piksela na ekranie. Każda z linii ekranu reprezentowana jest przez odpowiednią liczbę słów (jedno słowo odpowiada 16 pikselom), których liczba w trybie wyświetlania 320 punktów wynosi 20. Amiga ma możliwość wyświetlenia od 1 do 6 płaszczyzn bitowych. Co to oznacza w praktyce? Jeżeli wyświetlamy tylko jeden bitplan (spójrz na ilustrację nr 2), na ekranie możemy uzyskać tylko dwa kolory: kolor tła (bit ma stan logiczny 0 - piksel nie jest zapalany) i kolor punktu (bit ma stan logiczny 1 - piksel jest zapalany). Jeżeli chcemy uzyskać większą liczbę kolorów, musimy użyć kilku bitplanów, które „nałożone" na siebie dadzą nam odpowiednią kombinację bitów, na podstawie której będzie mógł być odczytany kolor punktu. Dzisiaj poznamy najprostszą i najważniejszą metodę uzyskiwania kolorów na ekranie.

Rejestry kolorów

Kolory na ekranie tworzone są przez wymieszanie w odpowiednich proporcjach trzech barw składowych: czerwonej (Red), zielonej (Green) i niebieskiej (Blue). Amiga dla określenia nasycenia każdej ze składowych potrzebuje czterech bitów, tak więc każda z barw może przyjmować wartości nasycenia od 0 do 15. Są trzy składowe, więc, jak łatwo obliczyć, możemy korzystać z palety 16*16*16=4096 kolorów. Nasz komputer ma 32 rejestry (tabela 3), które tworzą tzw. tabelę kolorów, i w których przechowywane są 12-bitowe wartości koloru. Wszystkie te rejestry są przeznaczone tylko do zapisu, tak więc nie możemy odczytać zapisanych w nich wartości. Budowa elementu tabeli jest prosta: bity 15-12 nie są używane, bity 11-8 są określają wartość składowej czerwonej, bity 7-4 - składowej zielonej, bity 3-0 - składowej niebieskiej.

Jak już wspomniałem, Amiga może wyświetlać od 1 do 6 bitplanów. Na ilustracji nr 3 pokazano, w jaki sposób Amiga, korzystając z czterech bitplanów, odczytuje kolor punktu i zapala go na ekranie. Pobierane są bity z kolejnych płaszczyzn bitowych i z nich tworzona jest liczba określająca, który kolor z tabeli będzie przypisany punktowi. I tak np. jeżeli bit w bitplanie 1 ma wartość logiczną 0, w bitplanie 2, 3 i 4 także, to uzyskana liczba równa będzie %0000 (zapisana w kodzie binarnym). Tak więc kolor przypisany punktowi będzie utworzony na podstawie wartości umieszczonej w rejestrze COLOR00 (adres rejestru=$DFF180 - jest to kolor tła).

W zależności od tego, z ilu korzystamy bitplanów, tyle kolorów mamy do dyspozycji:

Należy zwrócić uwagę, że tabela kolorów składa się jedynie z 32 rejestrów. Mogłoby to wskazywać na to, że maksymalnie możemy korzystać tylko z 5-ciu płaszczyzn bitowych, a tym samym maksymalna liczba kolorów na ekranie wynosiłaby 32. Tak nie jest. Amiga może obsługiwać 6 płaszczyzn bitowych, dając nam do dyspozycji 62 (tryb Extra Half-Bright) kolory z palety 4096-ciu lub 4096

Tabela 2

Liczba płaszczyzn	Liczba kolorów
1	2
2	4
3	8
4	16
5	32
6	64 i 4096

(tryb HAM) kolorów jednocześnie wyświetlonych na ekranie. O tych dwóch specjalnych trybach powiemy sobie w następnym odcinku naszego cyklu.

Play Field, czyli pole gry

Jest to nic innego, jak ciągły obszar pamięci, będący czymś w rodzaju ekranu graficznego. Na pole gry składają się kolejne płaszczyzny bitowe. Amiga daje nam różne możliwości wyświetlania pól gry: możemy wyświetlać od 2 do 4096 kolorów, obraz może składać się z dwóch niezależnych pól gry i możemy je płynnie przesuwać w obu kierunkach. Pole gry złożone jest z płaszczyzn bitowych, a wielkość jednej płaszczyzny zależy od używanej przez nas rozdzielczości.

I tak, przy najniższej rozdzielczości poziomej, która wynosi 320 pikseli, jedna linia będzie miała długość 20 słów (320/16=20), przy rozdzielczości 640 punktów długość linii wyniesie 40 słów (640/16). Możemy teraz obliczyć, ile pamięci zajmie nam obraz w najniższej rozdzielczości (320x256) w 4 kolorach (dwie płaszczyzny bitowe). W linii mamy 20 słów, a więc 40 bajtów, linii jest 256: 40*256=10240 bajtów w jednej płaszczyźnie bitowej. Mamy dwie płaszczyzny, więc wynik musimy pomnożyć przez 2: 10240*2=20480 bajtów (20kB) - tyle zajmie nam pamięć obrazu.

Rozmiar ekranu

Amiga pozwala nam na ustawienie dowolnej pozycji okna ekranu. Dlaczego musimy taką pozycję ustawić? Normalnie na ekranie nie jest wyświetlany cały

obraz. Jego widzialny obszar zaczyna się zwykle kilka linii i kolumn po przerwie wygaszania pionowego. Poza tym lampa kineskopowa nie jest prostokątna i ma zaokrąglone rogi. Obraz, który miałby wypełniać cały ekran, byłby niewidoczny lub zniekształcony w rogach.

Do ustawienia pozycji lewego górnego rogu ekranu służy rejestr DIWSTRT ($DFF08E). Położenie początkowe ograniczone jest do lewej górnej ćwiartki ekranu: bity od 15 do 8 określają położenie pionowe lewego górnego rogu okna wizyjnego, bity 7-0 określają położenie poziome. Obie wartości są więc zapisane w ośmiu bitach, przyjmując wartości od 0 do 255. Normalnie górny lewy róg ekranu ma położenie 129 pionowo i 41 poziomo, tak więc liczba zapisana w rejestrze DIWSTRT będzie równa $2981.

Pozycję dolnego prawego rogu ekranu ustawiamy w drugim rejestrze: DIWSTOP ($DFF090). Wartość zawarta w rejestrze wyznacza linię i kolumnę końca okna wizyjnego powiększoną o 1 (o jedną kolumnę i jedną linię za obrazem). Jeżeli ustawiamy poziomą pozycję końcową, która zapisana jest w rejestrze w dolnych ośmiu bitach (bity 7-0), musimy pamiętać, że komputer przyjmuje za zawsze ustawiony bit ósmy pozycji poziomej. Jeżeli wpiszemy do rejestru wartość dla położenia poziomego równą 5, to komputer uzna, że chodzi nam o położenie 261 (256+5=261). Wynika z tego, że współrzędne poziome końca okna wizyjnego mogą przyjmować wartości od 256 do 458.

Inaczej rozwiązano określanie pozycji pionowej końca okna (bity 15-8 rejestru). Bit 8 pozycji pionowej jest odwrotnością bitu 7 wpisanego do rejestru: jeżeli będzie on ustawiony, to bit 8 będzie wyzerowany i wartości współrzędnej będą w zakresie od 128 do 312. Jeżeli natomiast bit 7 będzie wyzerowany, to bit 8 zostanie automatycznie ustawiony i uzyskamy wartości od 256 do 312. Normalnie koniec okna wizyjnego ma współrzędne 449 poziomo i 297 pionowo. Tak więc wartość wpisana do rejestru DIWSTOP będzie równa $29C1.

Linia \ Kolumna	0	16	32	...	304
0	1-wsze słowo	2-gie słowo	3-cie słowo	...	20-te słowo
1	21-wsze słowo	22-gie słowo	23-cie słowo	...	40-te słowo
.	.	.	.	...	.
.	.	.	.	...	.
199	3980 słowo	3981 słowo	3982 słowo	...	4000 słowo



Okres wygaszania pionowego narzuca nam pewne ograniczenia co do wartości w rejestrach DIWSTRT i DIWSTOP. W pionie obszar widzialny ograniczony jest do linii z przedziału od 26 do 312 ($1A-$138). Pozycje poziome są dostępne począwszy od kolumny 107 ($6B).

Przy określaniu rozmiarów i położenia obrazu należy pamiętać, że rozdzielczość jest tu równa jednej linii rastra (pozycja pionowa) i jednemu pikselowi (pozycja pozioma) w niskiej rozdzielczości (320x256).

Po ustaleniu położenia i rozmiarów okna ekranu należy ustawić położenie początku i końca DMA płaszczyzn bitowych. Pozycji pionowej nie musimy ustawiać, DMA ekranu rozpoczyna i kończy w tej samej linii co okno wizyjne ustawione w DIWSTRT i DIWSTOP. Pozycję poziomą (cykl szyny DMA) wpisujemy do rejestrów: DDFSTRT ($DFF092 - dla pierwszej płaszczyzny bitowej) i DDFSTOP ($DFF094 - dla ostatniej płaszczyzny). Przy wyznaczaniu tych wartości korzystamy ze wzorów:

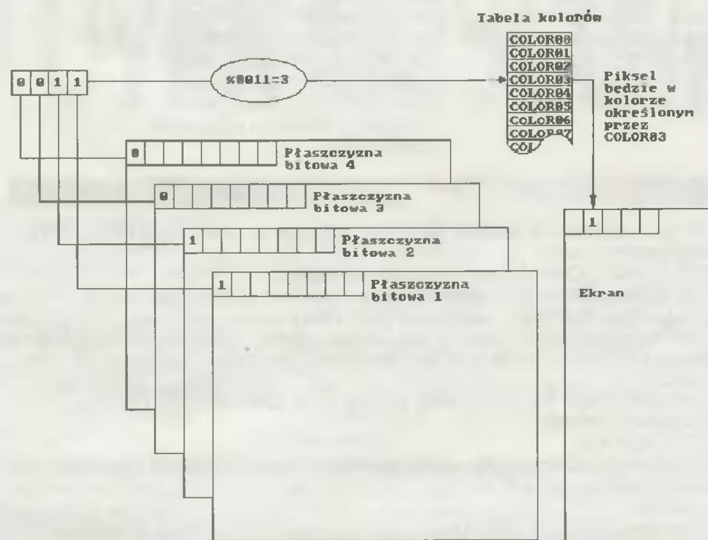
- w trybie niskiej rozdzielczości:
  DDFSTRT = (HStart/2-8.5) AND $FFF8
  DDFSTOP = DDFSTRT+(ilość pikseli w linii/2-8)
- w trybie wysokiej rozdzielczości:
  DDFSTRT = (HStart/2-4.5) AND $FFFC
  DDFSTOP = DDFSTRT+(ilość pikseli w linii/4-8)

HStart jest wartością poziomą początku okna ekranu.

W przypadku HStart=$81 i 320 pikseli w linii, wartość w rejestrze DDFSTRT będzie równa $38, a w DDFSTOP - $D0. Zaś przy wysokiej rozdzielczości (640 pikseli w linii) i HStart=$81 wartości będą równe odpowiednio: DDFSTRT=$3C, DDFSTOP=$D4.

Adresy map bitowych

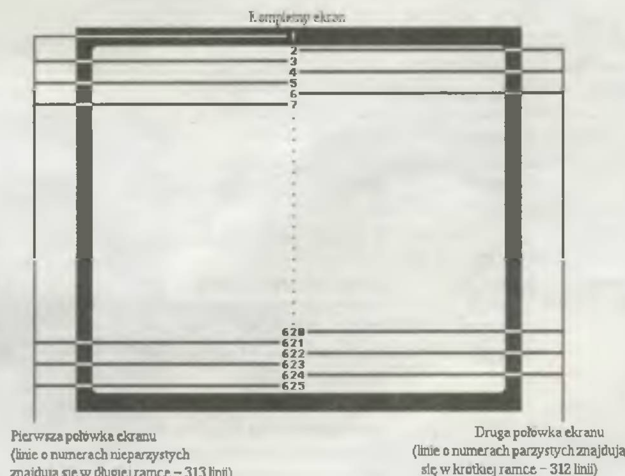
Amiga obsługuje maksymalnie sześć płaszczyzn bitowych. Adresy zawsze są wartościami zapisanymi w 32 bitach (Amiga korzysta z 19 bitów - płaszczyzna bitowa musi znajdować się w CHIP-RAM), a rejestry komputera są 16-to bitowe. Jest więc 12 rejestrów służących do ustawiania adresów map bitowych. Każda płaszczyzna bitowa posiada parę rejestrów. W jednym znajdują się górne 3 bity adresu płaszczyzny (bity 16-18), a w drugim dolne 16 bitów płaszczyzny (bity 0-15):




```
BPL1PTH   $DFF0E0 - adres płaszczyzny bitowej 1 (bity 16-18)
BPL1PTL   $DFF0E2 - adres płaszczyzny bitowej 1 (bity 1-15)
BPL2PTH   $DFF0E4 - adres płaszczyzny bitowej 2 (bity 16-18)
BPL2PTL   $DFF0E6 - adres płaszczyzny bitowej 2 (bity 1-15)
BPL3PTH   $DFF0E8 - adres płaszczyzny bitowej 3 (bity 16-18)
BPL3PTL   $DFF0EA - adres płaszczyzny bitowej 3 (bity 1-15)
BPL4PTH   $DFF0EC - adres płaszczyzny bitowej 4 (bity 16-18)
BPL4PTL   $DFF0EE - adres płaszczyzny bitowej 4 (bity 1-15)
BPL5PTH   $DFF0F0 - adres płaszczyzny bitowej 5 (bity 16-18)
BPL5PTL   $DFF0F2 - adres płaszczyzny bitowej 5 (bity 1-15)
BPL6PTH   $DFF0F4 - adres płaszczyzny bitowej 6 (bity 16-18)
BPL6PTL   $DFF0F6 - adres płaszczyzny bitowej 6 (bity 1-15)
```

Jak sterownik DMA wyświetla płaszczyznę bitową? DMA bitplanów pozostaje nieaktywne do momentu natrafienia na linię określoną w DIWSTRT. Następnie pobiera słowo danych dla płaszczyzn bitowych spod adresów umieszczonych w rejestrach BPLxPT. Dokonuje tego w momencie określonym przez wartość znajdującą się w rejestrze DDFSTRT. Po pobraniu słowa danych i umieszczeniu w odpowiednim rejestrze BPLxDAT zawartość rejestru adresu płaszczyzny bitowej zwiększana jest o 2 (przesunięcie do następnego słowa danych). Słowami umieszczonymi w BPLxDAT zajmuje się następnie kość DENISE - odczytuje bit po bicie, wybiera odpowiedni kolor na podstawie pobranych bitów i wysyła go na ekran. Po osiągnięciu pozycji określonej przez DDFSTOP, DMA czeka na DDFSTRT. Cały proces powtarza się aż do ostatniej linii zdefiniowanego okna ekranu, kiedy wiązka elektronów osiągnie pozycję zgodną z wartością w DIWSTOP. Teraz adresy w rejestrach BPLxPT będą wskazywać na pierwsze słowo za płaszczyzną bitową, dlatego też adresy muszą być odświeżane (ustawiane na nowo) po każdej ramce (co 1/50 sekundy). Robi to za nas procesor Copper, pod warunkiem, że skonstruujemy odpowiedni dla niego program, np.:

```
AdrBPxH  - adres płaszczyzny x (bity 16-18)
AdrBPxL  - adres płaszczyzny x (bity 0-15)
dc.w  $0E0,AdrBP1H          ;MOVE #AdrBP1H,$DFF0E0
```



Pierwsza połówka ekranu (linie o numerach nieparzystych znajdują się w długiej ramce – 313 linii)

Druga połówka ekranu (linie o numerach parzystych znajdują się w krótkiej ramce – 312 linii)

```
dc.w  $0E2,AdrBP1L      ;MOVE #AdrBP1L,$DFF0E2
dc.w  $0E4,AdrBP2H      ;MOVE #AdrBP2H,$DFF0E4
dc.w  $0E6,AdrBP2L      ;MOVE #AdrBP2L,$DFF0E6
dc.w  $0E8,AdrBP3H      ;MOVE #AdrBP3H,$DFF0E8
dc.w  $0EA,AdrBP3L      ;MOVE #AdrBP3L,$DFF0EA
dc.w  $FFFF,$FFFE       ;WAIT ($FF,$FE) - koniec
                        ;programu, zwanego popularnie
                        ;copper-listą
```

Rejestry sterujące

Amiga posiada trzy rejestry sterujące, służące do ustawienia odpowiednich parametrów obrazu:

BPLCON0 $DFF100 - ustawienie trybów pracy,
BPLCON1 $DFF102 - wartości dla przesuwania w poziomie,
BPLCON2 $DFF104 - rejestr sterujący priorytetami.

Tym razem poznamy jeden z nich, najważniejszy - BPLCON0. W tabeli nr 4 opisano przeznaczenie bitów w tym rejestrze, tu omówię je pod kątem praktyki:

HIRES
Ustawienie tego bitu spowoduje przełączenie rozdzielczości poziomej z 320 na 640 pikseli w linii.

BPUO-BPUO2
Trzybitowa liczba określająca, ile płaszczyzn bitowych będzie używanych (dozwolone wartości od 0 do 6).

COLOR
Tym bitem włączamy wyjście koloru Agnusa, mikser koloru będzie w stanie tworzyć kolorowy sygnał wizyjny. W przeciwnym razie sygnał będzie monochromatyczny.

Przeznaczenie pozostałych bitów poznamy w następnym odcinku naszego cyklu.

Obraz na ekran

Po przydługim i może trochę męczącym (ale bardzo potrzebnym) wstępie, spróbujemy, opierając się na podanych informacjach, wyświetlić na ekranie naszego monitora obraz, np. w dwóch kolorach przy rozdzielczości 640 pikseli na 256 linii. Jeżeli chcemy uzyskać obraz, musi zostać włączony kanał DMA dla płaszczyzn bitowych. Normalnie używamy także Coppera, dlatego jego kanał DMA także musi zostać włączony. Obowiązuje jeden podstawowy schemat tego typu czynności:

1. Wyłączamy kanały DMA, np.:
   MOVE.W #$01FF,$DFF096,
2. Ustawiamy adres copper-listy,
3. Wpisujemy do rejestrów odpowiednie wartości, ustawiamy: kolory, rozdzielczość, a także tryb pracy (HAM, DBPLF), priorytety, przesunięcie w poziomie (o tym będzie mowa w następnym odcinku naszego cyklu),
4. Ustawiamy w copper-liście adresy płaszczyzn bitowych,
5. Włączamy kanały DMA dla bitplanów i Coppera:
   MOVE.W #$8310,$DFF096.

Poniżej znajduje się krótki program, za pomocą którego możemy przeglądać pamięć naszego komputera. Ekran przesuwamy w dół i górę za pomocą strzałek sterujących kursorem, powrót następuje po wciśnięciu lewego przycisku myszy. Do uruchomienia programu potrzebny będzie Wam jakikolwiek asembler, na przykład Asm-One.

BARTOSZ SMAGA „Smuggler"

Tabela kolorów	
adres rejestru	nazwa rejestru
$DFF180	COLOR00
$DFF182	COLOR01
$DFF184	COLOR02
itd.	itd.
$DFF1BA	COLOR29
$DFF1BC	COLOR30
$DFF1BE	COLOR31

Tabela 3

Rejestr	BPLCON0 $100	
bit	Nazwa	Funkcja
15	HIRES	Włącznik trybu wysokiej rozdzielczości
14	BPU2	}
13	BPU1	} Liczba używanych płaszczyzn (od 0 do 6)
12	BPU0	}
11	HOMOD	Włącznik trybu HAM
10	DBPLF	Włącznik trybu podwójnego pola gry
9	COLOR	Kolor na wyjściu wizyjnym
8	GAUD	Włącznik Genlocka na kanale dźwiękowym
7-4	-	Niewykorzystane
3	LPEN	Aktywacja wejścia pióra świetlnego
2	LACE	Włącznik trybu INTERLACE
1	ERSY	Włącznik synchronizacji zewnętrznej
0	-	Niewykorzystane

Tabela 4

```
START:
   lea      COPPERLIST(pc),a0; adres cop.list
   move.w   #$01ff,$dff096    ; zamknięcie kanałów DMA
   move.l   a0,$dff080        ; ustawienie adresu cop.listy
   clr.w    $dff088           ; wyzerowanie rejestru COPJMP1
   move.w   #$2981,$dff08e    ; wartość dla DIWSTRT
   move.w   #$29c1,$dff090    ; wartość dla DIWSTOP
   move.w   #$3c,$dff092      ; wartość dla DDFSTRT
   move.w   #$d4,$dff094      ; wartość dla DDFSTOP
   move.w   #$000,$dff180     ; kolor tła
   move.w   #$f5f,$dff182     ; kolor punktów
   move.w   #$9200,$dff100    ; BPLCON0 - HIRES,COLOR,
                              ; 1 bitplane
   move.w   #0,$dff102        ; BPLCON1  }
   move.w   #0,$dff104        ; BPLCON2  } o nich w
   move.w   #0,$dff108        ; BPL1MOD  } następnym odcinku
   move.w   #0,$dff10a        ; BPL2MOD  }
   move.w   #$8380,$dff096    ; włączenie kanału DMA dla
                              ; Coppera i bitplanów
MOUSE:
   clr.l    d0
   move.b   $bfec01,d0        ; kod wciśniętego klawisza w D0
   cmp.b    #$65,d0           ; czy strzałka w dół
   beq.w    STRZALKADOL       ; tak w dół
   cmp.b    #$67,d0           ; czy strzałka w górę
   beq.w    STRZALKAGORA      ; tak w górę
   btst     #$6,$bfe001       ; oczekiwanie na LMB
   bne.b    MOUSE             ; jeżeli nie wciśnięty to skok
                              ; do etykiety MOUSE
WYJSCIE:
   move.l   $4,a6             ; EXECBASE do A6
   lea      GRlib(pc),a1      ; nazwa biblioteki w A1
   jsr      -408(a6)          ; otworzenie biblioteki
   move.l   d0,a0
   move.w   #$01ff,$dff096    ; wyłączenie kanałów DMA
   move.l   $26(a0),$dff080   ; wpisanie adresu systemowej
                              ; cop.listy
   clr.w    $dff088
   move.w   #$83ff,$dff096    ; włączenie wszystkich kanałów
                              ; DMA
   move.l   d0,a1             ; bazowy adres biblioteki w A1
   jsr      -414(a6)          ; zamknięcie biblioteki
   rts                        ; wyjście z programu
STRZALKADOL:               ; wciśnięto strzałkę do dołu
   lea      ADRBPL(pc),a0
   add.l    #80,(a0)       ; przesunięcie ekranu o jedną linię
                           ; w dół
                           ; (linia ma 80 bajtów długości)
   bra.w    WPISANIEADR
STRZALKAGORA:              ; wciśnięto strzałkę do góry
   lea      ADRBPL(pc),a0
   sub.l    #80,(a0)       ; przesunięcie ekranu o jedną linię
                           ; w górę
WPISANIEADR:        ; procedura wpisuje adres do copper-listy
   bsr      WAIT255
   lea      COPPERLIST(pc),a0
   move.l   ADRBPL(pc),d0
   move.w   d0,6(a0)      ; wpisanie dolnych 16 bitów adresu
   swap d0                ; zamiana bitów w rejestrze d0
   move.w   d0,2(a0)      ; wpisanie górnych 16 bitów adresu
   bra.w    MOUSE
WAIT255:                   ; oczekiwanie na koniec ramki
   cmpi.b   #$ff,$dff006 ; sprawdzenie położenia wiązki
   bne.w    WAIT255        ; elektronów (czy doszła do końca
                           ; ekranu)
   rts
ADRBPL: dc.l   0 ; tu przechowywany jest adres płaszczyzny
                 ; bitowej
GRlib:  dc.b    'graphics.library',0 ; nazwa biblioteki
     even                            ; instrukcja asemblera
                                     ; (parzysty adres)
COPPERLIST:                   ; początek copper-listy
    dc.w    $0e0,$0           ; adres płaszczyzny 1 (bity 16-18)
    dc.w    $0e2,$0           ; adres płaszczyzny 1 (bity 0-15)
    dc.w    $ffff,$fffe       ; instrukcja nielegalna
                              ; (koniec copperlisty)
```

TRACKBALL TKB-MT

Nie tak dawno polski rynek komputerowy zalała szybko zwiększająca się „populacja myszy". Obecnie mamy do czynienia z podobnym zjawiskiem w odniesieniu do trackball'i - pojawiają się coraz to nowe modele i to w sporych ilościach. Zasada działania trackball'a jest taka sama jak u myszy (układ opto-mechaniczny zamienia ruch obracającej się kuli na ruch wskaźnika) ale obsługa jest „postawiona na głowie" w sensie dosłownym. W trackball'u kulkę obracamy bezpośrednio palcami, natomiast całe urządzenie spoczywa nieruchomo.

NA POCZĄTKU BYŁ PAPIER...

Jako człowiek sumienny testowanie rozpocząłem od przejrzenia „papierów" dostarczonych razem z urządzeniem. Z miejsca rzucił mi się w oczy mały kolorowy folder reklamujący inne wyroby firmy Alfa Data. Nie miał on nic wspólnego z instalacją trackball'a, więc dokładne zapoznanie się z nim odłożyłem na później. Bardziej interesowałem się instrukcją obsługi. Wynika z niej, że ten model trackball'a współpracuje z Atari oraz podłączony do C-64 emuluje joystick. Wszystko zależy od ustawienia przełącznika na spodzie urządzenia. Niestety, z powodu braku wyżej wymienionych komputerów nie mogłem przekonać się o tym osobiście.

Instrukcja szczegółowo wyjaśnia jak podłączyć trackball'a i jak ustawić przełącznik do pracy z innymi niż Amiga komputerami. Ostatnim dokumentem dołączonym do zestawu jest karta gwarancyjna, i ta zrobiła na mnie duże wrażenie. Producent udziela gwarancji na poprawne działanie tego urządzenia aż na dwa lata. Dodatkowo karta gwarancyjna zawiera ankietę, po wysłaniu której nabywca urządzeń firmy Alfa Data będzie informowany na bieżąco o nowościach sprzedawanych przez firmę.

BUDOWA I WYKONANIE

Po przebrnięciu przez obowiązkową lekturę wyjąłem trackball'a z pudełka i podłączyłem do komputera. Od razu przypadł mi do gustu. Kolorystycznie dopasowany do naszej „przyjaciółki" jest co prawda dwa i pół raza większy od myszy, ale za to ponad trzy razy mniejszy od jej podkładki. Ponadto wyposażony jest w długi, półtorametrowy kabel. Szybko wgrałem Workbencha aby przekonać się czy TO w ogóle działa i rozpocząłem testowanie. Początkowo miałem trochę kłopotów z przyzwyczajeniem się do nowego urządzenia, co chwila łapałem się na tym, że chcę przesuwać je w całości zamiast obracać kulkę. Trackball TKB-MT jest tak zaprojektowany, że kulkę umieszczoną z lewej strony obraca się kciukiem, natomiast przyciski umieszczone w górnym prawym rogu powinno się naciskać palcami wskazującym i środkowym. Ale z powodu ich wielkości początkowo miałem z tym kłopoty. Niezależnie od tego jaki klawisz chciałem przycisnąć, naciskałem lewy.

W sumie przyciski wzbudziły we mnie mieszane uczucia. W przeciwieństwie do standardowej myszy są one bardzo miękkie i czułe, aż do przesady. Nawet niewielki nacisk wywołuje zadziałanie przycisku. Zdarzyło mi się zawiesić program naciskając lewy klawisz i przez przypadek razem z nim prawy. Czasem naciśnięcie, zwłaszcza prawego klawisza, można wywołać nawet przy mimowolnej zmianie pozycji dłoni.

Na pochwałę zasługuje umieszczenie na końcu lewego przycisku czegoś w rodzaju krawężnika. Dzięki niemu wiemy, gdzie kończy się lewy, a zaczyna prawy klawisz bez ciągłego spoglądania w jego kierunku. Po trzech dniach używania trackball'a zamiast myszy wypracowałem sobie odpowiednie ustawienie dłoni i pozbyłem się początkowego wrażenia, że ten model jest dla ludzi o dużych dłoniach i sprawności palców pianisty.

W AKCJI

Nadszedł więc czas na gruntowne próby. Na pierwszy ogień poszedł Deluxe Paint. Nie miałem żadnych problemów z rysowaniem z wyjątkiem bardzo długich lini, których z przyczyn technicznych nie udaje się narysować „jednym pociągnięciem pędzla". Obrót kulki z jednego skrajnego położenia w drugie pozwala na przesunięcie wskaźnika tylko do połowy ekranu.

Następnie na warsztat wziąłem moją ulubioną strzelaninę Battle Squadron. Z tą grą poradziłem sobie dopiero przy pomocy obu rąk. Jedną ręką nie dało rady. Zanim zdążyłem przełożyć kciuk, aby kontynuować ruch, byłem zniszczony. Szukając następnej gry do pomęczenia przypomniałem sobie, że zawsze, gdy grałem w Team Yankee, od ciągłego rozglądania się za nieprzyjacielem drętwiała mi ręka aż do łokcia. Z trackball'em było inaczej - nawet trzy scenariusze rozegrane pod rząd nie zmniejszyły mojej kondycji psychofizycznej.

Na koniec sprawdziłem, czy przypadkiem po przełączeniu trackball'a na tryb Atari/C-64 nie będzie emulowany joystick na Amidze. Niestety, moje podejrzenia się nie sprawdziły.

PODSUMOWANIE

Trackball TKB-MT sprawdził się w codziennej pracy z komputerem. Bez problemów obsługiwałem nim wszelkie programy użytkowe, z których korzystałem w ciągu dwóch tygodni testowania. Z żalem się z nim rozstawałem a ciarki mnie przechodziły, gdy pomyślałem o „drewnianych", w porównaniu z trackball'em, klawiszach myszy.

Zakup opisywanego urządzenia polecam każdemu, kto nie nabiera trwałych przyzwyczajeń manualnych i potrafi z łatwością „przerzucić się" z myszy na „kota"... (chyba ktoś kiedyś nazwał tak trackball'a?) W trackball'a powinni też zaopatrzyć się wszyscy cierpiący na brak miejsca na stole obok swojej mocno „okablowanej" i oprzyrządowanej Amigi.

PAWEŁ GALAS



DANE TECHNICZNE:
wymiary: 155x105x35 mm
waga: 300 g
rozdzielczość: 200 DPI trwałość mechanizmu: 100 mil (ok. 160 km)
trwałość przycisków: 1 mln kliknięć

ZALETY:
+ estetyczne wykonanie
+ długi przewód
+ dwuletnia gwarancja

WADY:
- zbyt czułe klawisze
- brak instrukcji w języku polskim oraz błędy i nieścisłości w tekstach obcojęzycznych

DYSTRYBUTOR: PROABIT, 05-500 Raszyn k. Warszawy, ul. Mickiewicza 14, tel. 56-08-91

CENA: ok 600.000 zł (59 DM)

AUDIOMASTER IV



Jakimi nowymi opcjami, w porównaniu do swojej poprzedniej wersji, dysponuje AUDiOMASTER IV? Czy możliwości tego programu są w związku z tym dużo większe, czy też tylko trochę?

Jak powszechnie wiadomo, Amiga ma ogromne możliwości muzyczne, które zawdzięcza czterem przetwornikom cyfrowo-analogowym. Jednak same przetworniki nie wystarczą, potrzebne są dane, które zostaną przez nie odtworzone. Aby stworzyć te dane, należy proces konwersji odwrócić, tzn. zastosować przetwornik analogowo-cyfrowy, zwany potocznie samplerem. Do jego obsługi stworzono wiele programów, jednak chyba najlepszy jest właśnie AUDIOMASTER IV. W stosunku do swego pierwowzoru (AUDIOMASTER I) różni się bardzo znacznie, praktycznie jest to zupełnie inny program. AUDIOMASTER IV różni się także od swojego poprzednika (AUDiOMASTER III) wieloma opcjami, które mogą zadziwić użytkownika.

PODOBIEŃSTWA...

Graficznie program jest łudząco podobny do AUDIOMASTERA III, właściwie można je rozróżnić tylko po napisie oznaczającym wersję, a znajdującym się w dolnej części ekranu. Opcja Sampler jest właściwie identyczna jak w poprzedniej wersji, różni się tylko zwiększonymi maksymalnymi częstotliwościami próbkowania, które wynoszą dla stereo 38 kHz, a dla mono - 55 kHz. Są to wartości „z górką" spełniające normy Hi-Fi.

Również menu Edit1 jest identyczne jak w AMIII. Użytkownik ma tu do dyspozycji opcje umożliwiające wycinanie, kopiowanie do bufora, zerowanie, wstawianie, odwracanie i zamianę miejscami kanałów (tylko dla stereo) fragmentów lub całości samplingu.

...I RÓŻNICE

Różnice zaczynają się od menu Edit2. Nowa opcja Adjust DC offset pozwala wyrównać amplitudę ujemnej i dodatniej połówki sygnału, co w praktyce oznacza brak trzasków i stuknięć w głośnikach przy włączaniu i wyłączaniu odtwarzania samplingu.

Dużo ciekawych opcji pojawiło się w menu Effects. Najciekawszymi z nich są: Digital Filter i Realtime Effects. Pierwsza z nich to po prostu filtr cyfrowy. Oferuje ona trzy podopcje: Boost - podbijanie określonego zakresu częstotliwości, Cut - wycinanie (tłumienie) określonego zakresu, i Pass - przepuszczanie (pozostawianie) określonego zakresu. Częstotliwość określa się przy pomocy dwóch suwaków, definiujących dwie wartości: dolną i górną częstotliwość (Lower Frequency, Upper Frequency). Trzeci suwak służy do ustawienia głębokości operacji filtrowania w procentach (0% - minimum, 100% - maksimum efektywności). Filtr taki umożliwia np. wycięcie szumów, podbicie słabo słyszalnych częstotliwości np. sopranów, lub przepuszczenie tylko środka pasma akustycznego (300 Hz - 3 kHz), co daje efekt „głosu przez telefon". Możliwości jest tu bardzo wiele, radzę spróbować.

Digital Filter jest bardzo ciekawą opcją, aczkolwiek nie pozbawioną wad: długie samplingi są przetwarzane czasem przez kilkanaście minut. Na szczęście podczas tej czynności program wyświetla czas pozostały do zakończenia operacji, co nie jest tak denerwujące, jak napis „Working, Please Wait..." występujący przy innych opcjach i w starszych wersjach AUDIOMASTERA.

Opcja Realtime Effects jest równie ciekawa. Mamy tu: echo w czasie rzeczywistym, opóźnienie w czasie rzeczywistym i zmianę częstotliwości w czasie rzeczywistym. Echo to po prostu kamera pogłosowa. Ustawiamy tu dwa parametry: Echo Rate, czyli liczbę odbić (powtórzeń) oraz Delay Rate, czyli czas wybrzmiewania. Ich działanie najłatwiej sprawdzić poprzez eksperymenty. Należy mówić do mikrofonu podłączonego do samplera jednocześnie zmieniając wartości parametrów. Przy małych wartościach Echo Rate nasz głos brzmi jak w pustej hali. Natomiast Delay Rate określa czas opóźnienia: gdy powiemy coś do mikrofonu, usłyszymy to natychmiast w jednym kanale, a po pewnym czasie w drugim - właśnie ten czas zależy od ustawienia Delay Rate.

Kolejna nowość to opcja zmiany częstotliwości - PitchBend. Jest to chyba najciekawszy efekt w programie. Umożliwia zmianę barwy głosu w czasie rzeczywistym. Ustawiamy tu tylko jeden parametr w procentach. 100 procent oznacza brak zmiany barwy - głos będzie naturalny. Dla wartości większych niż 100% (np. 150%) nasz głos zabrzmi jak głos Kaczora Donalda. Z kolei przy wartościach mniejszych (75%) zaczniemy mówić basem.

Kolejne menu nazwane Options zawiera opcje konfigurujące sampler, umożliwiające zmianę kolorów ekranu, włączenie bądź wyłączenie filtru, wybór parametru wyświetlanego w czasie odtwarzania samplingu (może to być czas, który upłynął od początku samplingu bądź pozycja odtwarzana w bajtach), możemy tu także włączyć ton oznajmiający zakończenie wykonywania jakiejś operacji oraz oversampling, czyli nadpróbkowanie (patrz słowniczek).

Ostatnie menu (Hi-Fi) zawiera opcje umożliwiające jak najwierniejsze odtworzenie samplingu oraz zapamiętanie samplingu na dysku w jakości Hi-Fi.

Na tym kończą się menu otwierane u góry ekranu (ang. pull-down). Reszta opcji dostępna jest na dole ekranu. Są to: funkcje odtwarzania, odtwarzania sekwencyjnego (także z powtórzeniami), powiększania wycinka samplingu, zapętlanie, odtwarzanie zaznaczonego fragmentu samplingu itp.

W programie zawarto jeszcze szereg bardzo przydatnych opcji, takich jak automatyczne rozpoczęcie próbkowania (czyli zapisu danych z samplera w pamięci) w momencie pojawienia się sygnału na wejściu samplera. Czułość ustawiamy w menu Options/Sampler Config. Gdy ustawimy za dużą czułość, to program może zareagować np. na szum dochodzący do wejścia.

Kolejną przydatną opcją jest możliwość zapamiętania utworzonego samplingu w różnych formatach np. IFF lub Sonix. Możemy także dokonać konwersji samplingu monofonicznego na stereofoniczny.

PODSUMOWANIE

Podsumowując możliwości AUDIOMASTERA IV należy stwierdzić, że program jest bardzo udany, oferuje mnóstwo ciekawych i przydatnych opcji (ciekawe czym jeszcze zaskoczą nas programiści w AUDIOMASTERZE V?), działa na wszystkich typach Amig. Aby jednak cokolwiek stworzyć, wymagane jest minimum 1 MB pamięci RAM, choć bardzo pożądana jest jej większa ilość, w przeciwnym razie przy niektórych opcjach możemy dość często oglądać napis „Not Enough Continuous Memory", co oznacza brak dostatecznej ilości wolnej pamięci (w postaci jednego, ciągłego obszaru). Na Amidze z 512 KB RAM nie działa większość ciekawych opcji, np. Echo Rate, a długość samplingu możliwego do stworzenia też pozostawia wiele do życzenia (ok. 2 s przy częstotliwości próbkowania 20 kHz).

Jeżeli nabrałeś, Czytelniku, ochoty na korzystanie z AUDIOMASTERA, a nie masz jeszcze samplera, to radzę sięgnąć do „C&A" nr 4/92 i spróbować wykonać opisany tam sampler.

JERZY DUDEK

AUDIOMASTER IV, 1991
RAMSCAN SOFTWARE under licence of AEGIS DEVELOPMENT

Nadpróbkowanie - ang. oversampling

Polega na próbkowaniu kompleksowego sygnału z częstotliwością czterokrotnie (poczwórny oversampling) lub ośmiokrotnie (ośmiokrotny oversampling) większą od częstotliwości próbkowania sygnału zapisanego. W wyniku tego otrzymujemy sygnały o innych pasmach częstotliwości. Pierwsze pasmo sygnału jest znacznie bardziej odsunięte od pasma częstotliwości sygnału analogowego. Daje to dużo większy odstęp od szumów.

OPTYMIZACJA DYSKIETEK

Podstawowym i najstarszym systemem zapisu plików na dyskietki jest w Amidze OFS. System ten ma jednak parę wad. Dobrze znane zgrzytanie i stukanie stacji dyskietek przy odczytywaniu danych spowodowane jest działalnością systemu operacyjnego, który przy zapisie „rozrzucił" je po całej dyskietce (bądź twardym dysku), a nie zapisał jako jedną całość. Również wolne wczytywanie katalogu dyskietki spowodowane jest tym, że nagłówki plików nie są umieszczone w jednym miejscu, lecz rozsiane po całej dyskietce.

Nowy system, tzw. FFS został znacznie ulepszony w stosunku do OFS, jednak nadal wczytywanie katalogów dyskietek i odczyt samych plików nie są najwygodniejsze i wzbudzają nasze obawy o stan głowicy w stacji.

W celu poprawienia sytuacji stworzono programy dające możliwość dokonania tzw. optymizacji dyskietek. Zamieszczona tabelka zawiera wyniki testów kilku najbardziej znanych optymizerów. Testy przeprowadzałem standardowej Amidze 500 (OS 2.04) z 2,5 MB pamięci RAM, oraz dyskietce „Workbench2.04".

Lp.	Program	Dysk-dysk	Jeden dysk	Uwagi
1	ReOrg V2.1	153 s.	145 s.	
2	B.A.D V4.13	270 s.	1476 s.	(!!!)
3	SinDiCo V1.31	150 s.	150 s.	
4	XCopyProfessional	-------	126 s.	bez weryfikacji
5	Fast Disk V1.9	156 s.	-------	
6	Flash Disk V 0.3	306 s.	580 s.	
7	Dos Control V4.0	-------	233 s.	
8	QBTools V1.5	-------	764 s.	

Dysk-dysk - *czas optymizacji dyskietki z jednego napędu na dyskietkę w drugim napędzie*
Jeden dysk - *czas optymizacji dyskietki w jednym napędzie*

XCopy Professional
Program znany zapewnie każdemu użytkownikowi Amigi. Nie każdy jednak wie o tym, że służy on nie tylko do kopiowania dyskietek, lecz ma także wbudowany dobry i bardzo szybki optymizer (radzę uważać na pirackie wersje programu, pewny jest tylko oryginał, i to od wersji 3.4 w górę). Szybkość (patrz tabelka) wynika między innymi z tego, iż zapis na dyskietkę dokonywany jest bez weryfikacji.

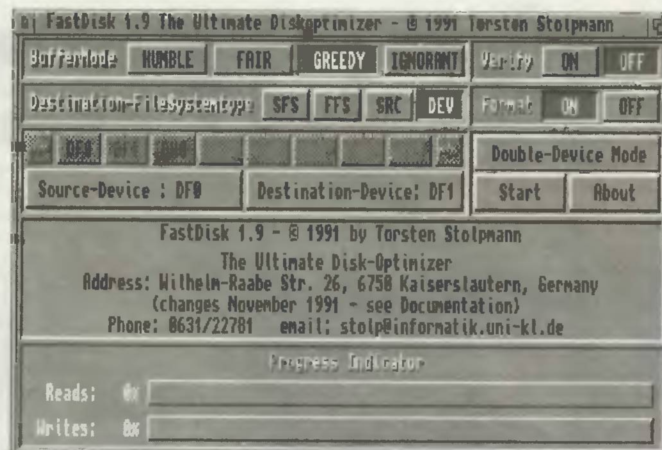
B.A.D.
Jeden z bardziej znanych programów optymizujących, zamieszczany bardzo często na dyskietkach Public Domain. Jest on jednak jednym z najwolniejszych optymizerów, jakie spotkałem. Przy pracy z dwiema stacjami dyskietek jeszcze jakoś sobie radzi, natomiast praca tylko z jedną stacją przeradza się w koszmar, proces optymizacji zajmuje mu 1476 sekund czyli ponad 25 minut (!). Dyskusyjnej urody jest również, przynajmniej według mnie, szata graficzna programu (kolory).

SinDiCo
Jeden z najszybszych optymizerów. Szybkość pracy programu nie zależy od liczby napędów, jakie są zainstalowane w systemie, gdyż wczytuje on do pamięci wszystkie informacje z dyskietki, a następnie, odpowiednio przetworzone, nagrywa na nią z powrotem. Do wad programu można zaliczyć to, że nie ma on żadnego interfejsu graficznego pozwalającego na łatwą i przyjemną komunikację z użytkownikiem. Pracuje tylko z poziomu CLI i wszystkie parametry należy podać przy uruchamianiu programu (sytuacja znana zapewne wszystkim mającym kontakt z programami takimi jak LhArc czy LhA).

QuarterBack Tools
Program narzędziowy przeznaczony głównie dla użytkowników twardego dysku, nadający się do pracy także z dyskietkami. Jedną z opcji programu jest możliwość optymizacji dysku, która nie należy jednak do jego mocnych stron. Pozwala na przeprowadzenie tej operacji wyłącznie na jednej stacji.

DosControl
Program narzędziowy (w stylu programów DiskMaster bądź Opus) pozwalający również na optymizację dyskietek. Dokonuje tej operacji tylko na jednym napędzie i niestety nie robi tego zbyt szybko.

Flash Disk
Bardzo popularny i często spotykany. Niestety, tak jak i w przypadku B.A.D.'a nie wiadomo, z czego wynika ta popularność. Program jest bardzo wolny (patrz tabelka), podczas optymizacji dyskietki przy wykorzystaniu tylko jednego napędu trzeba mieć przygotowaną także i drugą dyskietkę. Flash Disk wczytuje część informacji z dyskietki do pamięci, następnie prosi o zmianę jej na inną i dokonuje na niej zapisu, operację tą powtarza się 3-4 razy niezależnie od ilości dostępnej pamięci.



Fast Disk
Wykonany bardzo estetycznie, pozwala na optymizację dyskietki tylko z jednej stacji na drugą (według zapewnień autora optymizacja przy wykorzystaniu jednej stacji ma być dostępna w wersji 2.0 programu). Pozwala na zmianę wielkości bufora używanego przez program (cztery rodzaje), co wpływa na szybkość dokonywanych operacji (im większy bufor, tym szybsza praca).

ReOrg
Zdecydowanie najlepszy i najwygodniejszy z omawianych programów. Estetycznie wykonany, wysoko „idiotoodporny" (niełatwo daje się wprowadzić w błąd), zauważyć można troskę o użytkownika - wszystkie ryzykowne operacje wymagają potwierdzenia. Wykonuje on swoją pracę szybko i składnie. Pozwala także na symulację procesu optymizacji (dzięki czemu można się zapoznać ze strukturą dyskietki), jest także jednym z najmniej pamięciożernych optymizerów dla twardego dysku. Przy okazji jest także najszybszy.

Rozpowszechniane są dwie wersje programu:
- wersja 1.1 przeznaczona dla systemu 1.x,
- wersje 2.1 i 2.32 przeznaczone dla systemu 2.0 lub wyżej.

Wszystkie powyższe uwagi odnoszą sie głównie do wersji 2.1 programu, gdyż wersja 1.1 nie ma interfejsu graficznego pozwalającego na komunikację z użytkownikiem, a zarazem jest trochę wolniejsza niż 2.1. Według zapewnień autora udoskonalana będzie tylko wersja 2.1 (2.32) programu. Dla początkujących ReOrg (wersje 2.x) ma wbudowany tzw. „help mode", podający informację o każdej z funkcji programu. Program (w wersjach 1.1 i 2.1) możecie znaleźć na naszej dyskietce Public Domain nr 4 wraz z oryginalną dokumentacją.

HIGHTOWER

OFS - ang. Old File System - Stary System Plików
System zapisu plików na dyskietkach stosowany głownie w Amigach wyposażonych w Kickstart z serii 1.x, dostępny także w Amigach z Kickstartem 2.0 lub wyżej.
SFS - Standard File System - Zwykły System Plików
To samo, co OFS.
FFS - ang. Fast File System - Szybki System Plików
System zapisu plików stosowany głównie dla twardego dysku, a także w Amigach z Kickstartem 2.0 dla dyskietek, poprawiony w stosunku do OFS, umożliwia między innymi zwiększenie pojemności dyskietki (a także twardego dysku) o ok. 5% (czyli w przypadku dyskietki o ok. 40 KB).
Optymizacja - proces układający pliki na dyskietce (bądź dysku twardym) w taki sposób, aby nagłówki plików zgrupowane były w jednym miejscu, a same pliki zajmowały jeden nieprzerwany blok. Powoduje to szybsze wczytywanie się katalogu dyskietki i szybsze odczytywanie samych plików.

PROJECT X



PROJECT X

W roku 3505 na planecie RYXX zamieszkałej przez istoty ludzkie wybuchła wojna będąca zaczątkiem całej serii wojen pomiędzy ludźmi a stworami przypominającymi gady. Nazwano je Sauronami i opowiadano o nich niesamowite historie, które miały być przestrogą dla każdego, kto chciał stanąć z nimi oko w oko.

Chciwość Kodana, cesarza Sauronów, rosła z każdą chwilą. Do roku 3506 na planecie RYXX nie było już ani jednego człowieka na wolności. Cała galaktyka pogrążyła się w ciemności i smutku. Na okupowanej planecie Kodan postanowił wybudować więzienie, z którego nie było ucieczki. Tam też została uwięziona grupa rebeliantów. Sytuacja stała się krytyczna!

Pewnemu człowiekowi udało się jednak uciec ze straszliwej twierdzy. Tym człowiekiem był Sonnix - stary przewoźnik galaktyczny, doskonały pilot małych statków, które na szczęście ostały się jeszcze, ukryte przed wrogami daleko od RYXX. Sonnix postanowił zasiąść za sterami jednego z tych statków kierowany szlachetnym zamiarem: uratować pozostałych ludzi i zniszczyć Sauronów.

Był tylko jeden problem. Kodan dowiedział się o ucieczce Sonnixa i wysłał za nim olbrzymie eskadry myśliwców. Sonnix musiał więc najpierw zniszczyć wszystkie siły wroga...

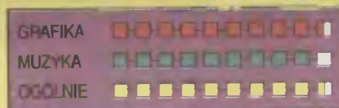
Gra polega na kierowaniu niewielkim stateczkiem lecącym poprzez galaktykę i strzelaniu do wszystkiego co się rusza albo strzela. Proste, a tak naprawdę bardzo trudne!

W spełnieniu misji może Ci pomóc komputer pokładowy, którego głos usłyszysz już po kilku sekundach od rozpoczęcia gry. Słuchaj uważnie wszystkich jego rad, gdyż w większości przypadków skorzystanie z nich okaże się lepsze od samodzielnie podejmowanych decyzji.

O PROJECT X mogę śmiało powiedzieć, że świat nie widział jeszcze takiej gry! Jest ona dziełem grupy TEAM 17, która robi chyba (jak do tej pory) najwspanialsze grafiki, animacje i oprawy dźwiękowe do gier (przez pierwsze pół godziny tylko siedziałem i gapiłem się w ekran!). Grę polecam właściwie wszystkim, a w szczególności "łamaczom joysticków". Mój ojciec, brat, kolega brata, ja, i chyba jeszcze pies - wszyscy jednocześnie chcieliśmy grać w PROJECT X! Kupcie, bo warto! A, i jeszcze jedno: uważajcie na meteory, są bardzo cwane!

OAK

FIRMA: TEAM 17
RODZAJ GRY: zręcznościowa
KOMPUTER: Amiga
WYMAGANIA: 1 MB RAM



PITSTOP 2

To gra dla wszystkich tych, którzy lubią szybką jazdę wyścigowymi samochodami. Na początku gry wybieramy:
- trasę,
- ilość graczy,
- ilość okrążeń,
- stopień trudności.

Podczas jazdy po dwupasmowej trasie, nasza formuła 1 zużywa zarówno paliwo, jak i opony. Na kołach pojawiają się kolorowe paski, które informują kierowcę o stopniu zużycia opon, a żółty skracający się pasek w dolnej części ekranu - o ilości pozostałego paliwa w baku. Po paliwo i na zmianę kół zjeżdża się do boksu. Tuż za metą, po lewej stronie pojawia się trzeci pas, na którym zapala się zielony kwadracik - to właśnie tu znajduje się boks. Jego obsługa składa się z dwóch ludzi, którzy niestety nie wykonują swoich zadań sami - Ty musisz nimi pokierować. Za pomocą białej kierownicy (sterowanej joystickiem) podprowadź człowieka ze szlauchem do samochodu aby nalał paliwo, a w międzyczasie drugiemu "każ" (joystickiem) zmienić koła.

Obsługa joysticka:
prawo, lewo - skręcanie
przód + FIRE - przyspieszanie
tył + FIRE - hamowanie
Klawisz RESTORE - rozpoczęcie gry od początku.

ROBERT KULIŚ



PITSTOP 2

RODZAJ GRY: zręcznościowa
KOMPUTER: C-64
WYMAGANIA: -



JAGUAR XJ220

Pewnego dnia, gdy już nic nie było do roboty, przypomniałem sobie, że przecież udało mi się zdobyć najnowszy program firmy CORE - grę JAGUAR XJ220. Postanowiłem w nią zagrać i...

Tak się zaczęła moja dwutygodniowa zabawa. JAGUAR XJ220 jest grą raczej sportową. Polega ona na kierowaniu najnowszym modelem jaguara, wozem szalenie szybkim, wygodnym, ekstrawaganckim, drogim, ładnym i z pewnością produkowanym tylko na zamówienie! Jednym słowem jazda takim „cackiem" jest marzeniem każdego prawdziwego mężczyzny! Na szczęście nie musisz być miliarderem, żeby się czymś takim przejechać, wystarczy, że masz Amigę.

Po pięknej i efektownej czołówce ukazuje się menu główne, o którym warto coś wiedzieć. Zacznij od pierwszego kwadracika w środkowym rzędzie (rysunek maski naszego jaguara XJ220). Gdy najedziesz myszą na ten kwadracik i klikniesz lewym guzikiem, w kwadracie pokażą się dwa samochody. Oznacza to, że gra ustawiona jest dla dwóch graczy. Jeżeli wybierzesz tę opcję, po prawej stronie uaktywnią się wcześniej „zamrożone" przez program inne ikony. Możesz wybierać między myszą a joystickiem, ręczną lub automatyczną skrzynią biegów, kontrolą szybkości za pomocą przycisku FIRE lub dźwigni joysticka. Możesz też ustalić liczbę okrążeń, jaką trzeba będzie przejechać w każdym etapie oraz wybrać rodzaj muzyczki przygrywającej podczas jazdy.

Szybkość jest tak zawrotna, że czasem nie widać mijanego pojazdu. Wszystko, co mknie przed Tobą lub obok Ciebie, musisz wyprzedzić i nie ważne, czy jest to lotus, porsche czy ferrari! Na drodze uważaj na słupki robót drogowych, bo jeśli się z nimi zderzysz, stracisz sporo czasu.

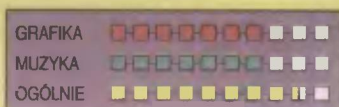
Podobno znana firma JAGUAR ogłosiła konkurs: ten, kto przejedzie wszystkie etapy w wymaganym czasie i zdobędzie najwięcej punktów, w nagrodę dostanie prawdziwego jaguara XJ220 na własność. Nie wiem, czy ta wiadomość jest prawdziwa, ale a nuż... Myślę, że warto się trochę pomęczyć i stanąć do konkursu!

Gra ta jest jakby kontynuacją starej, ale lubianej gry pt. LOTUS TURBO ESPRIT. Jednakże porównywać je byłoby conajmniej nietaktem wobec JAGUARA, który jest na pewno sto razy lepszy od wszystkich swoich starszych odpowiedników. Po-

zwolę sobie sądzić, iż JAGUAR XJ220 będzie niedługo grą przez wszystkich poszukiwaną. Proponuję więc przestać czytać, wyciągnąć z puszki chomikowane pieniądze i szybko pobiec zakupić tę grę, bo już jutro może jej nie starczyć dla wszystkich! Życzę połamania kierownicy!

OAK

FIRMA: CORE
RODZAJ GRY: symulacyjna, zręcznościowa
KOMPUTER: Amiga
WYMAGANIA: 1 MB RAM





JAGUAR XJ220

RAID OVER MOSCOW

Rozpoczyna się trzecia wojna światowa. Z pewnej bazy wojskowej w ZSRR startują cztery bombowce. Ich celem jest zbombardowanie jednej z ośmiu baz USA. Twoim zadaniem, jako obrońcy USA, jest skuteczny kontratak. Polega on na odnalezieniu i zniszczeniu wrogiej bazy wojskowej, z której wyleciały bombowce. Czasu jest niewiele, a więc, aby akcja powiodła się, musi zostać wykonana przed zniszczeniem Twej bazy przez samoloty wroga. Jeżeli nie uda Ci się tego dokonać, to niestety stracisz jedną ze swych baz.

Tak więc prowadzisz wojnę obronną, w której musisz zniszczyć trzy bazy normalne i jedną specjalną. Tę ostatnią atakujesz pod koniec gry. Znajduje się ona w Moskwie i jest najtrudniejsza do zniszczenia.

ETAP 1

Po obraniu stopnia trudności (klawisze F1, F2 i F3) komputer pokazuje wycinek kuli ziemskiej. Z rosyjskiej bazy, która przybrała biały kolor, startują bombowce. Tuż nad Europą, na orbicie okołoziemskiej, znajduje się potężny prom kosmiczny (na ekranie - niewielki obiekt koloru białego), na którego pokładzie znajdują się Twoje bojowe samoloty.

Po naciśnięciu spacji znajdziesz się wewnątrz promu. Jak widzisz, jest to hangar, w którym znajduje się Twój latający sprzęt wojskowy. Na pasie startowym stoi samolot, z którego skorzystasz w pierwszej kolejności. Musisz poczekać, aż pilot, który wszedł do hangaru, uruchomi silniki samolotu i zacznie nim kołować. Odpowiednimi ruchami joysticka ustaw samolot na pasie startowym tak, aby uzyskał on jak najmniejszą prędkość kołowania. Dziób samolotu musi być skierowany na właz wylotowy. Przytrzymując FIRE wznosisz samolot na pewną wysokość, następnie naciskasz F7 i wylatujesz.

Sterowanie samolotem w hangarze:

prawo,lewo - obracanie samolotu podczas kołowania,

przód - nadawanie lub wytracanie prędkości samolotu,

FIRE - wznoszenie samolotu,

F7 - otwarcie włazu wylotowego (tylko wtedy, gdy samolot znajduje się w powietrzu).

ETAP 2

Po pomyślnym opuszczeniu pokładu promu kosmicznego leć swym samolotem (na ekranie - biały migający punkt) do bazy, z której wystartowały wrogie maszyny.

ETAP 3

Teraz będziesz przelatywał tuż nad wrogą bazą. Musisz zniszczyć wszystko co się da (budynki, zbiorniki z ropą, czołgi, samochody itp.). Utrudnieniem tego etapu są rakiety, które dopiero od pewnego pułapu wysokości są w stanie trafić w Twój samolot (rakiety te samodzielnie namierzają cel). Musisz uważać na strzelający do Ciebie czołg, oraz śmigłowiec.

Po zestrzeleniu śmigłowca komputer przenosi Cię do najważniejszego punktu sowieckiej bazy. Przed Twym samolotem znajduje się pięć wież, które musisz zniszczyć. Najważniejsza, a zarazem największa, znajduje się w środku. Jest ona jakby sercem bazy. Jeśli ją zniszczysz, to baza przestanie istnieć. Dzięki wcześniejszemu zniszczeniu mniejszych wież, zyskasz więcej punktów. Nie jest to jednak takie proste. Swój samolot musisz ustawić przed wieżą na odpowiedniej wysokości tak, aby zmienił on swoją barwę z żółtej na fioletową. Dopiero z tej pozycji jesteś w stanie zniszczyć obraną wieżę. Utrudnieniem są działa przeciwlotnicze, umieszczone w każdej z wież. Ponadto musisz niszczyć samoloty wylatujące z lewej strony ekranu.

Wszystkie powyżej opisane etapy powtarzać się będą trzykrotnie podczas wykonywania misji (trzy bazy ZSRR). Ostatni atak, czyli na Moskwę, rozpoczyna się od drugiego etapu. Jego scenariusz jest nieco odmienny od poprzednich. Szczegóły starcia na placu Czerwonym oraz dalszy przebieg tej akcji pozostawiam w tajemnicy... zobacz sam, co się tam dzieje.

RAID... to jedna z lepszych gier zręcznościowych, jakie zostały napisane na C-64. Posiada ona dobrą grafikę i dość ciekawe efekty dźwiękowe. Przeznaczona jest dla gracza o dużej cierpliwości i wyobraźni. Trudna, ale bardzo wciągająca. Polecam i gwarantuję dobrą i ciekawą zabawę.



RAID OVER MOSCOW

ROBERT KULIŚ

FIRMA: ANTIRAM
RODZAJ GRY: zręcznościowa
KOMPUTER: C-64
WYMAGANIA: -



ADAM'S FAMILY

Któregoś dnia dzieci Adamsów zostały niedopilnowane i postanowiły wejść na strych. Pomysł ten był niewinny do czasu, kiedy to dzieci odnalazły starą księgę i zabrały się do czytania. Dzieci nic nie wiedziały o zaklęciu rzuconym na książkę, a raczej na złego ducha, który w niej mieszkał. Jak możemy się domyśleć, duch oczywiście uciekł z książki, a zaklęcie rzucił na dzieci. W końcu wyszło na to, że zły duch dowcipniś wydostał się wolność, a dzieci Adamsów zgubiły się we własnym domu!

Zatem wiesz już na czym polega gra: jesteś ojcem i szukasz własnych dzieci. W domu znajduje się mnóstwo różnych żywych rzeczy, które robią wszystko, by Ci przeszkodzić. Nie przejmuj się tym - stwory możesz zabijać, wystarczy, że na nie wskoczysz. Niestety nie możesz zlikwidować np. wirujących tasaków, gilotyny czy zwisających ostrz. Pamiętaj, że wszystko co się rusza, może Cię zniszczyć. Wiedz też, że dwa serduszka w lewym górnym rogu to Twoje życia.

Myślę, że nie ma co rozpisywać się więcej o tej grze. Dodam tylko, iż jest ona typowo rozrywkowa i raczej dla młodszych posiadaczy AMIGI, a ponadto ma doskonałą grafikę i muzykę.

OAK

FIRMA: OCEAN
RODZAJ GRY: przygodowa, zręcznościowa
KOMPUTER: Amiga
WYMAGANIA: wytrzymały joystick





ADAM'S FAMILY

C-64 CONTRA 800XL

AUDIATUR ET ALTERA PARS



Zatarg pomiędzy atarowcami i komodorowcami powinien, jako swego rodzaju curiosum, wejść na trwałe do historii informatyki. Przypadający na połowę lat osiemdziesiątych boom komputerów ośmiobitowych zbiegł się ze szczytowym nasileniem tych dwóch szowinizmów. Szczęściem, wraz z przemijaniem C-64 i małych Atari ten żenujący „cyrk" zaczął tracić na popularności i wydawało się, że sprawa przyschnie zupełnie, gdy „C&A" w numerze 6/92 podjęło temat od nowa. Mowa oczywiście o artykule „C-64 kontra Atari 800XL". Sama w sobie chęć porównania tych dwóch maszynek jest, wobec liczby nadchodzących listów, zupełnie zrozumiała. Wierzę również, że autor nie miał złych intencji, niemniej udało mu się sprawić, że w zespole bajtkowego klanu Atari zawrzało. Powodem są rzecz jasna znalezione w tekście niedopowiedzenia, nieścisłości, a nawet, o zgrozo, informacje nie mające pokrycia w rzeczywistości. Stwierdzenie, iż należy się nam małe sprostowanie, nie mija się chyba bardzo z prawdą.

PROCESOR

Ośmiobitowe Atari wyposażone są w procesor 6502 lub 6502C, przy czym pod tym drugim określeniem rozumiem 6502 z rozszerzoną listą rozkazów. Oba taktowane są częstotliwością 1,79 MHz (słownie: miliona siedmiuset dziewięćdziesięciu tysięcy Hertzów). W Commodore 64 montowane są procesory 6510 lub 8510 z zegarem 1 MHz.

PAMIĘĆ

Atari 800XL, 800XE i 65XE wysposażone są standardowo w 64 KB RAM, jakkolwiek 2 KB przysłonięte sprzętowymi rejestrami I/O nie są dostępne dla programisty. Maksymalny rozmiar pamięci RAM bez stosowania zewnętrznych kart rozszerzających wynosi 256 KB dla 800XL/65XE i 320 KB dla 130XE.

PORTY

Podstawowymi gniazdami 800XL są port szeregowy (standard Atari) i porty joysticków. Do tego pierwszego przyłącza się zwyczajowo stacje dysków (do czterech), magnetofon, modem, drukarkę. Porty joysticków mogą służyć jako stuprocentowe złącza I/O i oprócz joysticków, myszki, tabliczki graficznej, pióra świetlnego czy wiosełek (potencjometrów) komputer może przez nie obsłużyć również magnetofon, drukarkę, digitizer grafiki lub dźwięku, a nawet osiemdziesięciokolumnowy monitor - służy do tego specjalny interfejs XEP80. Do wyprowadzonej na zewnątrz modeli XL i większości XE szyny równoległej podłącza się kontroler twardego dysku (SCSI). Urządzenie to jest standardowo przewidziane w systemie operacyjnym, który potrafi je rozpoznać i obsługiwać.

GRAFIKA I DŹWIĘK

Maksymalną rozdzielczość ekranu Atari wynosi w STANDARDOWYM trybie graficznym 320*192 punktów, oprócz tego komputer ma dziesięć innych trybów grafiki o różnej rozdzielczości i liczbie kolorów wybieranych z palety 128 lub 256. Dostępnych jest sześć trybów znakowych. Manipulowanie wysokością (max. 238 linii) i szerokością (max. 384 punkty) obrazu, czy dowolne mieszanie trybów wyświetlania jest wyjątkowo proste i załatwia się przysłowiowym „jednym pokiem". Atari dysponuje również ośmioma „duszkami" (po cztery „duże" i „małe" lub pięć „dużych"), chociaż są one wyraźnie mniej wyrafinowane niż w C-64.

Atari XL/XE ma cztery, w pełni niezależne od siebie kanały dźwiękowe o skali 3,5 oktawy generujące standardowo sygnał o przebiegu prostokątnym. Można go zmienić przez wyłączenie (oddzielnie dla każdego z kanałów) dzielnika częstotliwości, co pozwala na bezpośrednie sterowanie położenia membrany głośnika. Każde dwa kanały można połączyć w jeden o skali 9 oktaw. Dzięki regulacji częstotliwości bazowej (15 kHz, 64 kHz i 1,79 MHz) teoretyczna rozpiętość częstotliwości wynosi od 0,12 Hz do ponad 126 kHz.

PAMIĘCI ZEWNĘTRZNE

Podstawową pamięcią zewnętrzną Atari są dyskietki elastyczne 5,25 cala. Stacje dysków (MFM) pracują w trzech gęstościach: pojedynczej 90 KB, średniej 130 KB i podwójnej 180 KB. Szybkość transmisji wynosi 19200 bitów/s (Commodore 1541 - 4000, format GCR). Stacja 1050 formatuje dyski w gęstości średniej (130 KB), lecz opracowane do niej sprzętowe rozszerzenia (US DOUBLER, Super Archiver, Turbo 1050, Top Drive i inne) umożliwiają korzystanie z podwójnej gęstości (180 KB) i przyspieszonej transmisji 70000 bitów/s lub więcej (max. 118000 bd). Najnowsza, dwustronna stacja Atari XF551 standardowo pozwala uzyskać 360 KB na dyskietce, zaś stacja TOMS 720 - 720 KB. Obydwie mogą pracować w trybie szybkiej transmisji Ultra Speed - 70000 bitów/s. Program zarządzający zbiorami dyskowymi (DOS) nie jest umieszczony w ROM-ie komputera czy stacji (jak w Commodore), lecz odczytywany z zewnątrz (na ogół z dyskietki), dzięki czemu użytkownik może dobrać sobie taki DOS, jaki mu odpowiada. Obecnie najlepszym DOS-em dla Atari jest SpartaDOS X wzorowany na systemie MS-DOS i umieszczony na module ROM (cartridge'u).

JĘZYKI PROGRAMOWANIA

Wobec fatalnych implementacji takich języków jak Pascal lub C podstawowymi translatorami używanymi przez programistów są Action, asemblery i BASIC. Uśmiech, który w tej chwili zagościł zapewne na obliczu niejednego z Czytelników nie jest jednak uzasadniony. Z moich doświadczeń wynika bowiem, że klony Atari BASIC są jednymi z najbardziej udanych dialektów tego języka w ogóle, a Turbo BASIC XL i BASIC XE są dodatkowo jednymi z najszybszych interpreterów dla komputerów ośmiobitowych. Jakkolwiek sam w sobie interpreter Atari BASIC jest, poprzez niewielką szybkość działania, narzędziem zaledwie znośnym, tzn. nadającym się jedynie do celów pomocniczych. Jednak jego użyteczność znacznie podnoszą kompilatory, m.in. znakomity MMG BASIC Compiler. Skompilowane nim programy niewiele ustępują szybkością programom napisanym w Action.

BASIC C-64 w opinii samych komodorowców jest „przykry", natomiast każdy atarowiec może pozazdrościć WARSAW BASIC-a. Język ten jest bogatszy od, mimo wszystko odeń szybszych, dialektów atarowskich (do dwóch razy - Turbo Basic XL).

CÓŻ JESZCZE

W toku zażartych dysput między atarowcami i komodorowcami obydwu stronnictwom umyka jakoś fakt istnienia takiej cechy Atari XL/XE o którą, mimo najszczerszych chęci nie da się posądzić Commodore'a. Cechą tą jest solidność wykonania. Objawia się ona na codzień zadziwiającą niekiedy idiotoodpornością, trwałością i znaczną bezawaryjnością komputerka. Na porządku dziennym są u atarowca zabiegi, które posiadacza C-64 przyprawiłyby o zawał: odłączanie i podłączanie stacji dysków, magnetofonu, drukarki, modemu, monitora (że o joystickach nie wspomnę) podczas pracy komputera. I jako żywo, nic się nie dzieje, gdyż atarowskie gniazda są na to obliczone. Mało tego - w systemie operacyjnym przewidziane są czynności podejmowane na wypadek wyjęcia, włożenia lub zamiany modułu ROM (catridge'a), rzecz jasna również podczas pracy komputera. Czym taki „numer" kończy się w przypadku Commodore 64 - wiedzą wszyscy.

PODSUMOWANIE

Pozostaje jeszcze tylko nie zgodzić się z konkluzją artykułu z „C&A" 6/92. Małym Atari, choć obchodzą w tym roku czternaste urodziny, daleko do całkowitej śmierci, a gdy taka nastąpi, to na pewno nie z powodu wątpliwej konkurencji C-64, lecz wspólnego „wroga": komputerów szesnasto- i trzydziestodwubitowych. Nawiasem mówiąc, Atari XE są, wbrew temu co sugeruje pan Łukasz Adamczyk, nadal produkowane - w Portugalii. Dziękuję za uwagę i oddaję głos Redakcji.

W imieniu atarowskiej frakcji ośmiobitowej spostrzeżenia i uwagi spisał:

KONRAD KOKOSZKIEWICZ

GROCH Z KAPUSTĄ

Na początku listopada 1992 roku na rynku oprogramowania ukazał się nowy produkt Fundacji Edukacji Technologicznej. Jest to dyskietka zawierająca 12 programów edukacyjno-użytkowych dla początkujących użytkowników Commodore 64. Do dyskietki dołączona jest dobrze opracowana instrukcja obsługi.

Autorem zbioru programów noszącego nazwę „GROCH Z KAPUSTĄ" jest pan Borys Żeszko, którego głównym założeniem było połączenie przyjemnego z pożytecznym. Przyjemną jest niewątpliwie prosta obsługa programów nie wymagająca właściwie żadnego przygotowania, no i samo obcowanie z komputerem, zaś jako pożyteczne należy uznać poznawanie i oswajanie się z możliwościami użytkowymi C-64 oraz oczywiście walory edukacyjne pakietu.

Omówię teraz pokrótce programy i ich możliwości. Przy nazwie, w nawiasie, podaję język, w jakim został napisany dany program.

ORTOGRAFIA (SIMON'S BASIC)
Program typowo edukacyjny zawierający 700 pytań testowych z zakresu polskiej pisowni. Szczególnie przydatny dla uczniów szkół podstawowych - po przerobieniu takiego materiału oceny dwójkowiczów z pewnością ulegną poprawie.

MINI-SLOW (SIMON'S BASIC)
Ten program napisany został z myślą o nauce języków obcych. Daje on możliwość utworzenia dowolnych słowników (np. polsko-angielskiego, polsko-niemieckiego itd.), które jako zbiory słówek mogą zostać zapisane na nośniku magnetycznym (taśma lub dyskietka). Prostą obsługę i dobrą komunikatywność programu zapewniają zawarte w nim funkcje:

1) odszukiwanie danego słowa - przez podanie jego pisowni,
2) odszukiwanie danego słowa - na podstawie jednego z jego znaczeń (np. niektóre angielskie słowa mają po kilka znaczeń w języku polskim),
3) przegląd zapisanych słów,
4) sortowanie słów wg kolejności alfabetycznej,
5) testy (sprawdzian ze znajomości zapisanych w słowniku wyrazów).

MINI-SLOW jest programem o naprawdę znaczących walorach edukacyjnych, kto nie wierzy, niechaj sam sprawdzi!

CHEMIA (SIMON'S BASIC)
Program edukacyjny pozwalający dobrze poznać i utrwalić wiadomości dotyczące tablicy okresowej pierwiastków chemicznych Mendelejewa. Warianty pracy programu - nauka, testy.

TOTO (SIMON'S BASIC)
Tej nazwy chyba nie muszę tłumaczyć. Jest to program dla miłośników gier liczbowych, za jego pomocą można nawet testować prawdziwe zakłady TOTOLOTKA.

WSD (SIMON'S BASIC)
Zadaniem tego programu jest tworzenie wykresów, słupków, diagramów. Podajemy wartości liczbowe, a komputer przedstawia nam graficzny obraz danych. Dzięki podobnemu programowi, lecz bardziej profesjonalnemu (konkretnie QUATTRO PRO V4.0 dla IBM), przedstawiliśmy w „C&A" nr 10/92 wyniki ankiety (str. 31-32). WSD może być niezłym wstępem do pracy z takimi programami.

KOSTKA (SIMON'S BASIC)
Program symulujący rzuty kostką do gry, zrealizowany w wielokolorowym trybie graficznym (multicolor), przez co znacznie zyskuje na atrakcyjności.

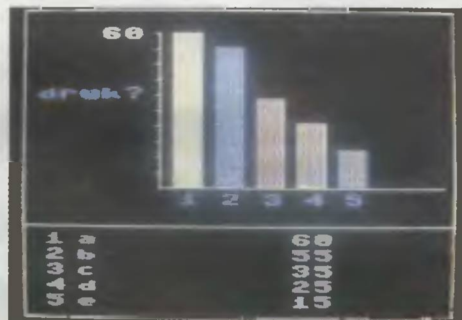
UNILOG-BZ (BASIC V2.0)
To uniwersalna baza danych, której zastosowanie uzależnione będzie od wyobraźni właściciela. Można dzięki niej stworzyć spis telefonów, adresów, książek, kaset, płyt, słowem czego tylko dusza zapragnie. Pojemność tworzonej bazy danych ograniczona jest jedynie ilością wolnej pamięci komputera. Program umożliwia:

- zapis danych,
- kasowanie danych,
- wyszukiwanie danych,
- wprowadzanie danych,
- poprawki danych,
- sortowanie danych wg porządku alfabetycznego,
- wydruk danych.

UNILOG-BZ daje użytkownikowi na poziomie szkoły podstawowej wystarczające pojęcie na temat możliwości baz danych w ogóle.

BIURO (BASIC V2.0)
Ten programik jest także swego rodzaju bazą danych. Służy on do gromadzenia i wyszukiwania informacji, którymi są oferty biura ogłoszeń. BIURO pozwala początkującemu użytkownikowi zapoznać się z możliwościami C-64 jako komputera biurowego, np. przy pracy w agencji reklamowej.

SKARBNIK (BASIC V2.0)
Mini arkusz kalkulacyjny pozwalający na prowadzenie małej księgowości. Program przeznaczony dla dzieci, bardzo prosty w obsłudze.



WSD

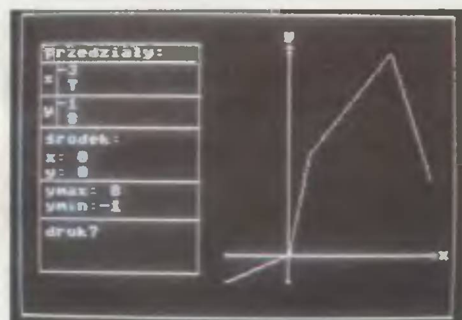
EDYTOREK (BASIC V2.0)
Ten edytorek tekstu służy do pisania i obróbki krótkich dokumentów. Oczywiście użytkownik ma możliwość zapisania owoców własnej pracy na taśmie lub dyskietce a także ich wydrukowania. EDYTOREK polecam każdemu początkującemu użytkownikowi C-64 jako przygotowanie do pracy z edytorami bardziej wyrafinowanymi, np. z POLSCRIPTEM 64 czy FONTMASTEREM.

UNI-TEST (BASIC V2.0)
Program służący do tworzenia testów z dowolnej dziedziny. Do programu autor dołączył trzy przykładowe testy:

- z geografii świata (300 pytań),
- z geografii Polski (150 pytań),
- ze słownika wyrazów obcych (dwie części, w sumie ok. 400 pytań).

Program ten serdecznie polecam nauczycielom posiadającym C-64, gdyż może on uprzyjemnić i ułatwić przygotowywanie pisemnych sprawdzianów.

KALENDARZ (BASIC V2.0)
Opracowanie kalendarza na kolejne 8 lat, czyli do roku 2000.



WSD

KALENDARZ

PODSUMUJMY

Choć przy przeglądaniu zawartości „GROCHU Z KAPUSTĄ" nie doznałem uczucia fascynacji, to uważam, że pakiet ten może być bardzo pożytecznym narzędziem edukacyjnym dla początkujących, młodocianych użytkowników C-64 i doskonale sprawdzi się jako zbiór programów wprowadzających w zaczarowany świat informatyki.

ROBERT KULIŚ

GROCH Z KAPUSTĄ, dyskietka + instrukcja
DYSTRYBUTOR I PRODUCENT: FUNDACJA EDUKACJI TECHNOLOGICZNEJ, ul. Burdzińskiego 5, 03-480 Warszawa, tel. 180-176

Z CZYM JADA SIĘ LOGO?

1. O czym jest mowa?

Logo to nie tylko nazwa znanego języka programowania. Logo, o którym mam zamiar pisać, pochodzi od starego, łacińskiego słowa logogram. Oznacza ono symbol lub skrót jakichś często używanych słów lub jakiejś nazwy. Jeżeli gorączkowo szukasz w pamięci jakiegoś przykładu, to zamknij pismo i popatrz na górną część okładki. Tak, tak. Te duże litery C&A razem z kolorowymi pasami tworzą logo. I dzięki temu, że litery są właśnie dokładnie takie a nie inne, że są pod nimi dwa pasy - niebieski i czerwony, już na pierwszy rzut oka wiesz, że masz do czynienia z „C&A" a nie np. z AMIGA WORLD. Jeśli chcesz obejrzeć sobie logo na ekranie, proponuję Ci uruchomienie dowolnego dema. Autorzy dem najczęściej umieszczają w nich dość duże ilości log (lóg? Oj, ciężko będzie odmieniać!). Jako treść owych lóg służy najczęściej nazwa grupy, do której należą autorzy programu lub trzyliterowy jej skrót (czyli TLS). Fotografia 1 przedstawia takie właśnie przykładowe logo, autorstwa Jetboy'a z grupy Cavern.

W rysowaniu loga pomocne są liczne programy, jak np. Centauri Logo Editor czy Multidrawer. Najczęściej zapisują one loga w pewien skuteczny, acz nie wszystkim znany sposób. Sposób ten jest z pewnością wart spopularyzowania wśród tych, którzy zaczynają dopiero pisanie własnych programów. Zwłaszcza dlatego, że - jak głosi stara mądrość włamywaczy - nie ma sensu wyważać otwartych drzwi i - o czym wiedzą podróżnicy i odkrywcy - powtórne odkrywanie Ameryki to robota raczej jałowa.

2. Przejdźmy do rzeczy!

Logo dość rzadko zajmuje więcej niż trzecią część ekranu. Osiem linijek - no, może dziesięć - pozwala grafikowi wyhasać się do woli i umieścić w logo co dusza zapragnie. W takiej sytuacji koderzy, którzy odbierali od grafików efekty ich pracy, doszli do wniosku, że marnotrastwem miejsca na dysku i w pamięci byłoby nagrywanie całego ekranu w postaci rysunku w wysokiej rozdzielczości (w trybie HI-RES byłoby to 8-9 kB, a w wielokolorowym - 10!). Poszukując jakiegoś mniej rozrzutnego i wygodniejszego sposobu zapisu loga koderzy użyli po prostu trybu tekstowego!

Zapytają co niektórzy: Jakże? Przecież tryb tekstowy to literki tylko! Czasem jeszcze znaczki jakoweś, z których wszelako nic ładnego wyjść nie może! (Tak między nami, to ostatnia teza jest dość dyskusyjna, ale dla dobra naszych rozważań przejdźmy nad nią do tzw. porządku.) Znakomici owi adwersarze zapominają jednak, że aby pokazać na ekranie literki, komputer musi mieć w pamięci jakieś ich odwzorowanie. Wiemy, że istnieje możliwość wyperswadowania komputerowi pobierania kształtu czcionek z ROM i zastąpienia tego wzorca jakąś własną czcionką, choćby w celu dodania języczka przy „ę". Komputer jednak jest głupi - mało tego! - jest potwornie głupi. Jeśli wstawimy mu w odpowiednie miejsce pamięci kółeczko i poinformujemy, że tak wygląda litera „A", to od tej pory za każdym razem zamiast niej pojawiać się będzie właśnie owo kółeczko. Dobre, co?

Z tego właśnie korzystamy nagrywając logo. Najpierw musimy „rozpisać" je na czcionkę i ekran. Powstał w tym celu specjalny algorytm, który bierze kolejne fragmenty rysunku, sprawdza, czy pokrywają się z poprzednimi i, w zależności od tego, zapisują nowy fragment jako kolejną literę alfabetu lub wstawiają na jego miejsce któryś z poprzednich znaków. Jak wygląda taka" rozpiska" ekranu pokazuje fotografia 2 - widzimy, że ogólne zarysy przypominają logo z fotografii 1.

Potem nagrywamy nasze dziełko na dysk. Najlepiej robić to za pomocą

Fot. 2





Fot. 1

dwóch plików. W jednym znajduje się nasz własny generator znaków, w drugim - rozkład tych znaków na ekranie. Możliwość taką posiada np. wspominany już Centauri Logo Editor - wystarczy wybrać opcję „Save screen + font”. Pliki te nazywa się najczęściej: „F NAZWARYSUNKU” - generator znaków (ang. font - czcionka, znak) i „S NAZWARYSUNKU” - rozkład na ekranie (screen).

3. Jak to wyświetlić?

Pozostaje jeszcze tylko mały problem techniczny: jak zmieniać komórkę $d018 (53272), by móc dowolnie wyświetlać logo? Przyjmijmy, że VIC widzi pierwsze 16 kB (boż tylko tyle może). Generator znaków ma wielkość 2 kB. 0000 - 07ff odpada. Za dużo problemów ze stroną zerową i przeróżnymi wektorami systemu. 0800 - 0fff dobre, jeśli nie używasz BASIC-a. 1000 - 17ff i 1800 - 1fff odpada. Ten obszar jest maskowany przez ROM od adresu $d000 do $dfff. 2000 - 27ff, 2800 - 2fff, 3000 - 37ff i 3800 - 3fff dobre. W te obszary możesz swobodnie, bez nerwów wstawiać swoje generatory znaków (za generator znaków odpowiedzialna jest młodsza połowa komórki $d018 (53272) czyli bity 0 - 3).

Tabela 1

Generatory znaków		
Obszar pamięci	HEX	DEC
0000 - 07ff	$01	000
0800 - 0fff	$03	003
1000 - 17ff	$05	005
1800 - 1fff	$07	007
2000 - 27ff	$09	009
2800 - 2fff	$0a	011
3000 - 37ff	$0c	013
3800 - 3fff	$0f	015

Tabelka 1 przedstawia zestawienie obszarów pamięci, do których można wpisać generator znaków i odpowiadających im wartości, które należy wstawić do młodszej połowy komórki $d018 (53272).

Oprócz samego generatora musimy jeszcze określić rozstawienie znaków na ekranie. Ekran zajmuje niecały 1 kB. Mamy więc dwukrotnie więcej możliwości, ale... 0000 - 03ff odpada. 0400 - 07ff jest w sumie niezłe, ale musimy się liczyć z tym, że po każdym resecie czy RUNSTOP/RESTORE zawartość tego obszaru jest czyszczona. 0800 - 0bff bywa używane, ale nie jest wygodne, bo nawet jeśli nie używamy BASIC-a, po każdym resecie czyszczone są trzy pierwsze komórki. 0c00 - 0fff dobre, a nawet niezłe. 1000 - 1fff całkiem odpada (jak wyżej). 2000 - 3fff dobre. Zestawienie obszarów pamięci z odpowiadającymi im wartościami do wstawienia do starszej połowy $d018 (53272) podaję w tabeli 2.

Tabela 2

Rozpiski ekranów		
Obszar pamięci	HEX	DEC
0000 - 03ff	$00	000
0400 - 07ff	$10	016
0800 - 0bff	$20	032
0c00 - 0fff	$30	048
1000 - 13ff	$40	064
1400 - 17ff	$50	080
1800 - 1bff	$60	096
1c00 - 1fff	$70	112
2000 - 23ff	$80	128
2400 - 27ff	$90	144
2800 - 2bff	$a0	160
2c00 - 2fff	$b0	176
3000 - 33ff	$c0	192
3400 - 37ff	$d0	208
3800 - 3bff	$e0	224
3c00 - 3fff	$f0	240

Teraz musimy wybrać sobie odpowiednie obszary pamięci. Dla przykładu wstawmy generator znaków do obszaru zaczynającego się od $2000 (8192) a „rozpiskę” ekranu od $0c00 (3072). Patrzymy do tabeli 1. $2000 to $8 (8). Teraz patrzymy do tabeli 2. Dla $0c00 odnajdujemy wartość $30 (48). Jeśli chcemy otrzymać liczbę, którą wstawimy do komórki $d018 (53272), musimy dodać to, co wzięliśmy z tabeli 1 do tego, co było w tabeli 2. Czyli szesnastkowo 38, co przekłada się na dziesiętne 56.

4. Kurtyna w górę!

Nadszedł czas na pokazanie naszego logo! Najprościej będzie wpisać:

```
POKE 53272, 56
```

Jak widzimy, metoda skuteczna. Ale od czegóż są na tym świecie programiści? Skomplikować potrafią najprostsze! Żeby nie odstawać od tego stereotypu napisałem specjalnie dla Was program, którego zadaniem jest zapytanie o nazwę rysunku, potem władowanie tego rysunku z dysku, pokazanie go na ekranie i porót do zwykłego ekranu po naciśnięciu spacji. To by było na dziś.

Wasz specyalny korespondent,

BARTEK KACHNIARZ

```
400 readn$,po,k
405 fora=1to6
410 reada(a):b=b+a(a)
415 ifa(a)<0then470
420 next
425 read c
430 ifc<>bthenprint"blad w
    linii";nl+1000:stop
435 b=0
440 nl=nl+5
445 fora=1to6
450 pokepo,a(a)
455 po=po+1
460 next
465 goto405
470 print"program "n$:print"
    uruchomienie:sys36864":new
999 datamalarz, 36864 , 37033
1000 data 032,129,255,032,132,255,835
1005 data 169,147,160,144,032,030,682
1010 data 171,162,000,032,207,255,827
1015 data 201,013,240,006,157,133,750
1020 data 144,232,208,243,169,070,1066
1025 data 141,131,144,169,008,170,763
1030 data 160,000,032,186,255,169,802
1035 data 015,162,131,160,144,032,644
1040 data 189,255,169,000,170,160,943
1045 data 032,032,213,255,169,083,784
1050 data 141,131,144,169,008,170,763
1055 data 160,000,032,186,255,169,802
1060 data 015,162,131,160,144,032,644
1065 data 189,255,169,000,170,160,943
1070 data 012,032,213,255,169,000,681
1075 data 141,032,208,141,033,208,763
1080 data 169,056,141,024,208,141,739
1085 data 022,208,169,064,197,203,863
1090 data 240,252,169,008,141,022,832
1095 data 208,169,021,141,024,208,771
1100 data 162,014,169,160,157,132,794
1105 data 144,202,208,250,096,073,973
1110 data 032,160,160,160,160,160,832
1115 data 160,160,160,160,160,160,960
1120 data 160,160,160,147,080,079,786
1125 data 068,065,074,032,084,089,412
1130 data 084,085,076,032,079,066,422
1135 data 082,065,090,075,065,013,390
1140 data 000,251,000,000,239,251,741
1145 data -1
1150 rem linie data przez ign-datamaker
```

JAK NAPISAĆ WŁASNE DEMO

cz. 2

Witajcie w drugiej części naszego cyklu. Zanim zabierzemy się do pracy, musimy poznać trochę słownictwa związanego z demami, co ułatwi i uprości zrozumienie omawianych dalej zagadnień. No to zaczynamy.

AUTOMODYFIKACJE to technika programowania polegająca na tym, że program, w zależności od różnych czynników, sam się modyfikuje w trakcie działania. Technika ta jest bardzo często używana w demach, gdyż pozwala na dość znaczne skrócenie czasu wykonywania programu.

CYKL (po angielsku: cycle). Cykl procesora jest to jednostka (wynosząca w przypadku procesora 6502 (6510) ok. jedną milionową część sekundy), którą mierzy się czas wykonywania rozkazów przez dany mikroprocesor. Podczas jednego cyklu wiązka elektronów tworząca obraz na ekranie przebiega odległość równą ośmiu punktom (ang. pixels) obrazu wysokiej rozdzielczości. Najkrótszy rozkaz (NOP) wykonywany jest w ciągu dwóch cykli. Znajomość liczby cykli potrzebnej do wykonania każdego rozkazu jest potrzebna do wyliczania czasu wykonywania procedur, a także do cyklowania.

CYKLOWANIE (po angielsku: cycling) to czynność polegająca na takim doborze opóźnień, aby wszystkie zmiany rejestrów następowały w odpowiednim miejscu ekranu. Jest to dosyć żmudne, ale czasami wręcz konieczne, np. przy otwieraniu bocznej ramki. Jeżeli w demie, na ekranie początki rastrów „szarpią", to mówimy, że program jest niedocyklowany.

CZAS RASTROWY (po angielsku: rastertime). To pojęcie ma bardzo duże znaczenie. Ekran jest odświeżany co 1/50 sekundy, a właściwie dzieje się to cały czas, tylko co 1/50 sekundy wiązka elektronów przechodzi przez to samo miejsce. Ponieważ ekran jest wyświetlany równolegle z pracą procesora, to można powiedzieć, że każdemu cyklowi odpowiada kawałek rastra. Zatem długość wykonywania możemy mierzyć w cyklach, albo w rastrach. Jest to o tyle ważne, że do uzyskania płynnych efektów musimy wykonywać odpowiednie procedury co 1/50 sekundy. Tak więc suma czasów wykonywania wszystkich procedur (muzyka, efekty, oczekiwanie na wciśnięcie spacji itp.) musi być mniejsza od czasu potrzebnego na wyświetlenie jednego ekranu (obrazu).

CZĘŚĆ (po angielsku: part). Jest to pojedynczy program, z którego składa się demo. W przypadku dem jednoplikowych kilka części jest połączonych w jeden program i możemy je uruchamiać tylko w określonej kolejności. W przypadku dem dyskowych (pod warunkiem, że nie używają one track-loader'ów) części nagrane są najczęściej jedna za drugą i możemy je uruchamiać niezależnie od siebie. Inną nazwą jest INTRO, ale odnosi się ona bardziej do „reklamówek" grup przed magazynami, demami lub użytkami.

DIGITALIZACJA to dźwięki przetworzone z postaci analogowej na zapis cyfrowy, możliwy do odtworzenia przez komputer.

LOGO jest to sposób przedstawienia rysunku za pomocą odpowiednio zdefiniowanego generatora znaków oraz mapy ich ustawienia (patrz artykuł pt. „Z czym jada się logo?" w tym numerze „C&A"). Technikę tę stosuje się zwykle przy przedstawianiu rysunków, które mają dużo powtarzających się wielokrotnie elementów - oszczędza się znacznie pamięć.

MODUŁ (ang. module) to player (czyli taki samogrający program; patrz niżej) albo nazwa przystawki (ang. cartridge) instalowanej do EXPANSION PORT.

NIBBEL (ang. nibble) jest to półbajt. 4 młodsze bity w bajcie to młodszy nibbel, a 4 starsze bity - starszy nibbel. Np. w bajcie #$ea, #$e to starszy nibbel, a #$a - młodszy. Pamięć koloru ($d800 - $dbff) składa się wyłącznie z młodszych nibbli.

NIEMOŻLIWE to słowo, które każdy koder powinien wykreślić ze swego słownika. O wielu rzeczach mówiono kiedyś, że są niemożliwe do zrobienia, a później wszyscy nauczyli się je robić i wydają się one obecnie czymś zupełnie zwyczajnym (ot, choćby taki SIDEBORDER, czyli umieszczanie grafiki na bocznych ramkach).

PIXEL jest to najmniejszy element obrazu. Na ekranie wysokiej rozdzielczości mamy do dyspozycji 320 pixeli w poziomie i 200 w pionie. W przypadku grafiki w multikolorze pixele są dwa razy większe w poziomie i do dyspozycji mamy rozdzielczość



160x200. Słowo pixel ma oczywiście swój polski odpowiednik - punkt, lecz w środowisku „demomanów" przyjęło się mówić pixel, a ja nie widzę powodu, by odchodzić od tego zwyczaju.

PROCEDURA GRAJĄCA (po angielsku: player) jest to procedura, która pobiera dane określające kolejne nuty i ewentualnie zmienia niektóre (wszystkie) parametry dźwięku, czyli po prostu gra melodię. Najczęściej trzeba ją (procedurę) odtwarzać co cykl rastra.

RASTER (po angielsku: raster). Przyjrzyj się z bliska monitorowi (ekranowi telewizora). Zauważyłeś, że ekran składa się z wąziutkich paseczków. Te właśnie paski to rastry. Inne znaczenie tego słowa to nazwa efektu, w którym każdy raster może przyjmować inny kolor.

SCROLL jest to przewijający się na ekranie napis - jeden z najbardziej podstawowych i najczęściej używanych w demach efektów.

WYCIĄGANIE (rypanie, prucie) jest to proceder polegający na ekstrakcji z programów pewnych ich części np. grafiki, muzyki, kodu, aby później użyć ich w swoich produkcjach. O ile używanie wyciągniętej muzyki, oczywiście z podaniem jej autora, nie budzi żadnych kontrowersji, to używanie cudzego kodu, albo grafiki jest rzeczą haniebną.

Do naszej nauki potrzebne będzie nam kilka pomocy naukowych. Po pierwsze jakiś moduł (najlepiej Action Replay V4.0 i wyższe, albo przynajmniej Final II lub III). Jest to bardzo pomocne urządzenie zwłaszcza, gdy chcemy wyciągać muzykę, lub podpatrywać „jak oni to zrobili?". Poza tym przydałby się jakiś asembler, najlepiej, gdyby był to TurboAssembler V5.0, ponieważ wszystkie programy będą pisane przy jego pomocy.

Niezbędna jest też znajomość kodu maszynowego, bo bez tego nawet nie ma co marzyć o pisaniu dem (polecam kurs „Asembler 6502" prowadzony na łamach „C&A" przez Bartka Kachniarza). Musicie też nauczyć się systemów dwójkowego, dziesiętnego (ten już chyba znasz) i szesnastkowego oraz przeliczania liczb pomiędzy nimi. Nie będziemy uczyć się tego w tym miejscu, gdyż temat ten był już wielokrotnie przerabiany we wszystkich magazynach komputerowych. Mam nadzieję, że do czasu ukazania się następnego numeru uda się Wam zdobyć potrzebny sprzęt i oprogramowanie.

RAFAŁ PIASEK

ASEMBLER 6502 (cz. 7)

SKOKI - KOMPUTER ŻABKĄ

Jak zapewne spostrzegłeś, komputer ma już prostą umiejętność wypisywania głupot (czasem i mądrości, ale dużo, dużo rzadziej) na swoim ekranie. Czyli odpowiednie procedury już „siedzą" gdzieś we wnętrzu naszej maszyny. Najlepszym miejscem jest ku temu pamięć ROM, która to potrafi przechowywać swą zawartość niezależnie od tego, czy jest zasilana prądem, czy też nie. Dodatkową jej zaletą jest fakt, że nie da się zmienić jej zawartości.

No, ale dość tych głupot, przejdźmy do konkretów. Żeby wydrukować na ekranie jakąś literę trzeba nam było najpierw wstawić ją do akumulatora rozkazem LDA a potem wyrzucić na ekran jakimś innym rozkazem (STA, ale to pomińmy). Dużym ułatwieniem naszej pracy byłby, po pobraniu numeru litery, skok do jakiejś procedury (ach, słodkie języki wyższego rzędu i ich procedury!) bez przejmowania się, cóż mądrego jest właściwie w tych procedurach, byle jeno robiły, co chcemy (czyli drukowały litery; jeszcze, ale już niedługo - nie tylko).

W ROM C-64 istnieje procedura, która nazywa się tajemniczo „CHROUT". Skrzaty mówią, że przy układaniu jej nazwy chodziło głównie o to, by nie dało się jej wymówić. A oznacza ona CHaRacter OUTput, czyli - wypchnięcie znaku. Służyć może do różnych celów - do wydrukowania danego znaku na drukarce, zapisania na dysku, albo (i o to właśnie chodzi) - na ekranie. Wystarczy jedynie wstawić znak do akumulatora (LDA) i skoczyć...

MONITOR	ASEMBLER
	*=10000
A2710 LDA #$01	LDA #$01
A2712 JSR $ffd2	JSR $ffd2
A2715 RTS	RTS

Krótkie, proste, ale skuteczne. Mamy „A". Jeśli pomyślisz chwilę, zaraz się domyślisz, jak tą metodą drukować dłuższe wyrazy, czy całe zdania. Przy czym ciekawe jest, że wcale nie interesuje nas, jak to się dzieje - litera po prostu się pokazuje i już. Łatwo zauważyć, że działamy tutaj metodą taką samą, jak np. w BASIC-u czy PASCAL-u - jeżeli pewna część programu musi być powtarzana więcej razy, prościej jest odwoływać się do niej kiedy trzeba, a nie wpisywać przy każdej konieczności. Dzięki temu możliwe staje się osławione programowanie strukturalne. Swoją drogą, jeżeli nie byłoby ono możliwe w asemblerze, jak można by wyobrazić sobie taką możliwość w jakimkolwiek innym języku? Asembler to podstawa. WSZYSTKO, co można zrobić np. w BASIC-u - zostało już wcześniej zrobione w asemblerze, tylko że nie przez nas, lecz przez autorów interpretera.

Rozkaz JSR może więc nam służyć, jeżeli chcemy z jakiegokolwiek miejsca programu skoczyć do innego. Dodatkową zaletą tego rozkazu jest fakt, że w pewnym miejscu pamięci, zwanym STOSem przechowuje adres, do którego powinien wrócić po wykonaniu procedury. Powrót następuje automatycznie, po napotkaniu rozkazu RTS. Jeżeli procesor trafi na ten rozkaz, to powraca do głównego biegu programu.

Innym „skaczacym" rozkazem jest JMP (JuMP). Jego cechą jest to, że skacze bezpośrenio pod wskazany adres i nie zostawia żadnych znaków szczególnych (takich jak JSR na stosie), by można było go po nich wyśledzić. Jeżeli skoczymy gdzieś poprzez JMP, nie istnieje możliwość powrotu do poprzedniej częsci programu. Rozkaz ten działa więc tak samo jak instrukcja GOTO w BASIC-u.

Jeszcze winien jestem wyjaśnienia w sprawie dalszego uproszczenia drukowania tekstów na ekranie. Otóż w pewnym miejscu w pamięci ROM istnieje procedura, która ma za zadanie wyświetlenie zadanego tekstu. Ot, i wszystko. Nie musimy już w tym celu konstruować pętli, pobierać kolejnych znaków itd. Wszystko zrobili już za nas autorzy systemu operacyjnego. Trzeba tylko:

a) wiedzieć gdzie jest ta procedura,

b) umieć zlecić komputerowi jej wykonanie, czego właśnie mam zamiar Was nauczyć.

Na początek muszę Wam przekazać, jak komputer zapisuje w pamięci dłuższe adresy. Robi to za pomocą tzw. starszych i młodszych bajtów. Aby wyliczyć jakiś adres mając starszy i młodszy bajt, trzeba ten starszy przemnożyć przez 256 i do wyniku dodać młodszy. Brzmi to może nieco zawile, ale nie zapominajmy, że myślimy ciągle systemem dziesiętnym (ach, gdyby tak Bóg stworzył człowieka z szesnastoma palcami!), komputer zaś dwójkowym, którego skróconym zapisem jest system szesnastkowy. Dlatego też rozłożenie liczby dwubajtowej (czyli większej niż 255, $ff) jest w systemie szesnastkowym proste, w dziesiętnym zaś - już nie tak bardzo. Dla liczby $1234 starszy bajt (z angielska MSB - More Significant Byte) wynosi $12, zaś młodszy (LSB - Less Significant Byte) - $34. Liczbę dzielimy więc po prostu na dwie części. Pierwsze dwie cyfry to bajt starszy, dwie ostatnie - młodszy. Zaś w systemie dziesiętnym rozpiska liczby 12345 wygląda tak: 12345 dzielimy na 256 (tu ukłon w stronę red. Kalkulatora), wynik: 48.222656. Od tego odrzucamy część po przecinku, zostaje więc 48. I to jest starszy bajt. Przechodzimy więc do młodszego. Pomnóżmy 48 przez 256, wyjdzie 12288. Teraz musimy jeszcze od naszej początkowej liczby (12345) odjąć to nieszczęsne 12288. Mi wyszło 57. Ten etap był zresztą najbardziej dramatyczny. Stary, blisko pięciocentymetrowej (!) grubości Texas Instruments odmówił współpracy i musiałem szukać czegoś innego. Sprawdźmy jednak, czy wynik jest prawdziwy. 48*256+57=12345. Wszystko się więc zgadza. Dla porządku przedstawię jeszcze algorytmy w BASIC-u:

MSB=INT(AD/256)
LSB=AD-256*MSB

Na pewno przydadzą się nie raz. Do czego jednak były potrzebne w tym przypadku? Do poinformowania komputera o adresie, od którego zaczyna się tekst do wyświetlenia na ekranie. Wpiszmy teraz jakiś tekst do pamięci, od adresu - niech będzie $3000, czyli 12288, posługując się metodą przedstawioną przeze mnie w poprzednim odcinku. Tym razem jednak koniecznie wstaw na końcu znaczek „@" o kodzie $00, 0. Gotowe? Jeśli tak, to wpisz:

MONITOR	ASEMBLER
	*=$2710
A2710 LDY #$30	LDY #$30
A2712 LDA #$00	LDA #$00
A2714 JSR $ab1e	JSR $ab1e
A2717 RTS	RTS

...i uruchom to jak zwykle - SYS 10000, G2710, czy strzałka w lewo, 3 i S. Teraz na ekranie pojawił się cały potrzebny Ci tekst, bez bólu, szybko i skutecznie. Jeżeli masz ochotę tekst ten jeszcze jakoś sformatować, zmienić kolory, czy co tam jeszcze - skorzystaj z tabeli kodów ASCII znajdującej się gdzieś pod koniec instrukcji obsługi. Jak widać, rolę starszego i młodszego bajtu wzięły na siebie dwa rejestry procesora - jako starszy wystąpił rejestr Y, jako młodszy - akumulator. Jest więc, jak obiecałem - umiemy wydrukować tekst za pomocą jednego rozkazu.

A teraz miła niespodzianka dla użytkowników TurboAssemblera. Nie wiem, czy metoda ta zadziała w innych programach tego typu, ja sprawdziłem ją jeszcze w Magic Assemblerze. Wpisz (tym razem monitorowcy mogą np. zamknąć oczy):

```
ASEMBLER

*=$2710
LDY #>reklama
LDA #<reklama
JSR $ab1e
RTS

reklama .TEXT „C&A - TWOI DOSTAWCY
        NARKOTYKOW"
```

Działa? I to jak! Od tej chwili możesz w ten sposób ułatwić sobie działania na młodszych i starszych bajtach tym właśnie prostym sposobem, o czym zawiadamia ostrzący przy szkwałach

BARTŁOMIEJ I. KACHNIARZ

Wielu z Was wiele o nich słyszało. Prawie każdy przynajmniej raz je widział. Ale nie wszyscy dokładnie wiedzą...

CO TO SĄ SPRAJTY?

Na dobry początek: sprajty (inaczej duszki, z angielska sprites) to małe obrazki, które możemy kontrolować niezależnie od ekranu głównego. Poza tym, że są niezależne, obrazki te mogą się wyrzynać, rzucać widelcami, biegać, obracać i robić mnóstwo innych rzeczy, które przyjdą nam do głowy. Musimy jednak wiedzieć, jak to właściwie zrobić. Zacznijmy więc od podstaw.

Kształt sprajta jest chyba najważniejszą jego cechą. Jeśli nie opiszemy komputerowi właściwego kształtu naszego pomysłu, zamiast międzygalaktycznego superbohatera możemy ujrzeć dobrze znaną kaszanę, którą nadaje TVP po zakończeniu emisji programu. Klasyczny, jednobarwny sprajt to mozaika stworzona z 504 zapalonych lub zgaszonych punktów, ułożonych w prostokąt 24 na 21 punktów. Punkty w sprajcie ułożone są nieco inaczej niż punkty na ekranie graficznym. Na każdą bowiem komórkę ekranu przypada ich dokładnie 8. Pierwsza komórka sprajta to lewa część jego najwyższej linijki. Druga komórka leży od niej trochę na prawo - pośrodku linijki, trzecia zaś - na prawym jej skraju. I to już cała górna linijka. Druga linijka uporządkowana jest identycznie: od lewej do prawej. Tak samo wygląda to we wszystkich 21 liniach (patrz rys. 1).

Aby uzyskać odpowiedni kształt, zapalamy lub gasimy poszczególne bity komórek. Dlatego też dobrze byłoby napisać program, który pobierze z linii DATA dane kształtu w postaci np. kropek (.) i gwiazdek (*), zer (0) i jedynek (1), literek „A" i ....B", czy co tam jeszcze przyjdzie nam do głowy, a następnie, przetrawiwszy to na zrozumiałe dla komputera liczby, wstawi we właściwe miejsce do pamięci. Z punktu widzenia naszej wygody najlepiej byłoby w każdej linii DATA dać dokładnie 24 znaki, dokładnie tyle co w linijce sprajta. Przyjmijmy więc, że mamy już taką oto linię:

```
900 DATA ...*..***...*.*....**.**
```

Musimy najpierw rozdzielić ją na trzy części po osiem punktów:

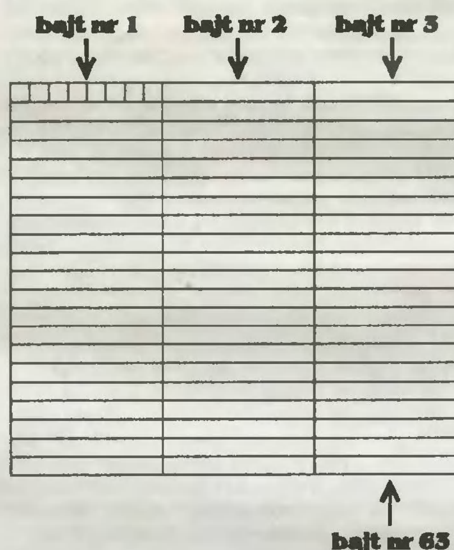
```
1: ...*..**
2: *...*.*.
3: ...**.**
```

... a następnie, wiedząc o tym, że gwiazdka to bity zapalone, a kropka - zgaszone, zapalić odpowiednie bity w bajtach. Osiem kropek musi się zmieścić w jednym bajcie, więc każdej kropce odpowiada jeden bit. Niestety, BASIC C-64 nie pozwala nam na bezpośrednie korzystanie z układu dwójkowego, więc będziemy musieli skorzystać z pośrednictwa układu dziesiętnego.

Wartości, jakie można przechować w bajcie to liczby od 0 do 255. Zerowy bit (pierwszy z prawej) ma wartość dziesiętną 1. Pierwszy - 2, drugi - 4, każdy kolejny - dwukrotnie od poprzedniego większą (8, 16, 32, 64 i 128). Tak więc wartość danego bitu wyliczyć możemy z wzoru:

$W=2^{NB}$

gdzie W to wartość dziesiętna, a NB - to numer naszego bitu. Jeśli któreś z pól pierwszego zestawu będzie gwiazdką, będziemy musieli zapalić odpowiadający mu bit. Tylko w taki sposób możemy dokładnie przekazać naszej maszynce, o co właściwie chodzi. Popatrzmy na kolejne bity: zerowy (prawy!) musi być zapalony, bo na jego polu postawiliśmy gwiazdkę, czyli 1. Pierwszy też. Do wartości bitu dodać musimy 2 (w sumie już 3). Drugi i trzeci - to kropki, więc bity pozostawiamy zgaszone. Czwarty - znowu gwiazdka, do wartości bitu dodajemy 2^4 czyli 16. Razem 19. Bity piąty, szósty i siódmy są puste. Wartością, którą wstawiamy do pierwszej komórki sprajta jest więc 19. Jeśli nie jest dla Ciebie jeszcze wystarczająco jasne o co tu biega, popróbuj na dwóch pozostałych bajtach.



Rys. 1

Na szczęście, nie musimy tego żmudnego zabiegu powtarzać 63 razy (bo tyle komórek zajmuje wzór sprajta w pamięci), by przygotować jakąkolwiek postać na ekranie. Bowiem zamiast pracować krótko, ale wiele razy, lepiej będzie zrobić to dłużej, ale za to raz na zawsze i przygotować sobie program, liczący to za nas. Po prostu złożymy kilka pętli, które pobiorą z linii DATA ciągi gwiazdek i kropek, przetrawią je na liczby i wstawią je do...

No właśnie: DOKĄD? Do tej pory ciągle pisałem - do pamięci. Pamięć jest jednak długa i... no, tu może przesadziłem, bardzo szeroka to ona jednak nie jest... W każdym razie miejsca jest akurat tyle, by dało się wepchnąć wzór sprajta w miejsce, w którym komputer najmniej się go spodziewa. A przecież żeby komputer mógł wykonywać nasze rozkazy, trzeba mu je najpierw wydać. Jak więc przechowujemy w pamięci matryce sprajtów?

Każdy wzór ma 63 bajty. Dla równego rachunku dodaje się jeden, niewykorzystany bajt, więc cały sprajt ma ich już 64. Dzięki temu sprajta możemy umieścić w każdym miejscu pamięci, byleby numer pierwszej komórki zajętej przez wzór był bez reszty podzielny przez 64. Miejsca te to np. 64, 128, 172, 704, 832, 896, 1024 i inne. Weźmy jednak pod uwagę, że - przy zwykłym ustawieniu wszystkich wektorów - adresy powyżej 1024 są raczej mało dostępne - zajmuje je w całości pamięć ekranu i program w BASIC-u. Poniżej tego adresu pamięć także eksploatowana jest dość intensywnie. Projektanci komputera przewidzieli jednak kilka dziur na nasze matryce. Dlatego też wygodnie będzie korzystać jedynie ze sprajtów w komórkach 704 - 767, 832 - 895 i 896 - 959 (przy czym dwa ostatnie położenia raczej bym odradzał, jeśli masz zamiar korzystać z magnetofonu).

Mamy więc, na dobry początek trzy miejsca w pamięci. Ich numery jako numery dziur dla sprajtów to 11 (704/64), 13 (832/64) i 14 (896/64). Jeżeli chcemy więc wpisać matrycę naszego sprajta od adresu 704, liczba odpowiadająca jego położeniu w pamięci będzie wynosiła 11. Poczucie przyzwoitości nakazywałoby gdzieś tę liczbę wstawić. Jako że sprajtów możemy mieć 8, przygotowano dla nich 8 komórek: 2040 - 2047. UWAGA! WAŻNE! Komórki te to ostatnie 8 lokacji ekranu tekstowego. Jeśli wektory ekranu będą wskazywać inne miejsce niż 1024, te adresy także się zmienią. Pamiętajmy więc, że numery dziur dla sprajtów siedzą w komórkach 1016 - 1023 licząc od początku ekranu! Na razie korzystajmy jednak ze zwykłej konfiguracji pamięci. Wstukajmy informację, że sprajt nr 0 ma wzór w dziurze nr 11:

```
POKE 2040, 11
```

Od tej pory, jeżeli każemy wyświetlić sprajta nr 0, na ekranie pokaże się zawartość komórek 704 - 767. I o to od początku nam chyba chodziło. Teraz musimy jeszcze tylko wiedzieć JAK to wyświetlić.

W pamięci układu VIC znajduje się rejestr 53269, odpowiedzialny za pojawianie się sprajtów na ekranie. Sprajtów jest osiem, bitów w bajcie też osiem, działa więc tu ten sam mechanizm, co w przypadku wpisywania do pamięci matrycy znaków. Chcemy nakazać pojawienie się sprajta nr 0 - zapalamy bit nr 0. Najprościej będzie uczynić to instrukcją:

```
POKE 53269, 1
```

Jeżeli jednak są już jakieś inne sprajty są na ekranie, napiszmy:

```
POKE 53269, PEEK (53269) OR 1
```

Dzięki temu nie zlikwidujemy pozostałych wstawiając nowy. Po naciśnięciu RETURN czeka nas jednak mały zawód: na ekranie nie pojawiło się nic nowego. Bardziej niecierpliwi zapewne rzucą w tym momencie wszystko w diabły, wyzywając mnie od niekompetentnych głupców. Resztę jednak informuję, że oprócz nakazania wyświetlenia musimy jeszcze wydać instrukcje co do miejsca, w którym sprajt ma się pokazać. W momencie włączenia komputera wszytkie sprajty mają ustawione te współrzędne na punkt 0,0, który ma to do siebie, że - niestety - cały jest przykryty obwódką. Współrzędne

sprajta nr 0 to: 53248 (współrzędna X) i 53249 (współrzędna Y). Następnie parami (X i Y) pou-kładane są współrzędne położeń pozostałych siedmiu sprajtów. Wpiszmy więc we współrzędne sprajta nr 0 takie wartości, by już na pewno cały wylazł spod obwódki:

POKE 53248, 100: POKE 53249, 100

Ciągle nic? To było do przewidzenia: komórki 704 - 767 w zasadzie powinny być puste. Zapełnijmy je więc treścią. Najprościej będzie wpisać tam, co elektrony na procesor przyniosą:

FOR A=704 TO 767:POKE A,INT(RND (1)*255):NEXT

O proszę, już coś się pokazuje! Dla uproszczenia nazwijmy to sztuką nowoczesną albo wręcz: obrazami namalowanymi przez komputer. Jeśli jednak chcemy mieć własne sprajty, zbierzmy wszystkie informacje, które do tej pory podałem i napiszmy program, który umieści nam na ekranie sprajta. Ale uwaga! Wcale niekoniecznie musisz bezkrytycznie przepisać to, co podam w listingu. Znacznie lepiej byłoby, gdybyś doszedł do tego sam...

```
200 rem ***********************
205 rem * co to jest sprajt? *
210 rem ***********************
215 rem * bartek kachniarz *
220 rem ***********************
225 poke 2040,11
230 poke 53269,1
235 poke 53248,100
240 poke 53249,100
245 for y=0 to 20
250 read sp$
255 for x=0 to 2
260 for b=0 to 7
265 t$=mid$(sp$,b+8*x+1,1)
270 if t$="*" then wb=wb+2^(7-b)
275 next
280 poke 704+y*3+z,wb
285 wb=0:z=z+1
290 next:z=0:next
295 end
300 data .....****.....*******...
305 data ..*******.....*******...
310 data .********....********...
315 data .****...**..***.****....
320 data ****...**..****.****....
325 data ****...*.**.****.****...
330 data ****...****.*********...
335 data .****......*********....
340 data .**************..****...
345 data ..*************...***...
350 data .....*********...****...
355 data ........................
360 data **********************..
365 data ****************........
370 data ************............
375 data *******.................
380 data ***.....................
385 data **......................
390 data *.......................
395 data *.......................
400 data
```

I już gotowy sprajt jest na ekranie. Jeżeli zechcesz, możesz zmieniać jego położenie odpowiednio regulując zawartość komórek 53248 (poziom) i 53249 (pion).

Jeszcze tylko jeden szczegół: w poziomie ekran jest szerszy niż 255 punktów. Powoduje to, że pozycji poziomej sprajta nie da się jednoznacznie określić ośmioma tylko bitami. Dodatkowy, najstarszy bit jest w komórce 53264. Jeśli go zapalisz, odpowiadający mu sprajt przesunięty zostanie w prawo o 256 punktów. Jeżeli chcesz uzyskać płynność ruchu, nie zapo-mnij wyzerować komórki 53248.

Mam nadzieję, że artykuł ten rozwieje większość Waszych wątpliwości dotyczących sprajtów.

BARTEK KACHNIARZ

PO PROSTU AMIGA

Na rynku wydawniczym pojawiają się coraz to nowe pozycje poświęcone Amidze. Objaw bardzo pozytywny: informacji nigdy nie za wiele. Jedną z książek, jakie ukazały się w ostatnim czasie, jest „Po prostu Amiga" panów Andrzeja Bobka i Bartosza Smagi.

W książce tej można wyróżnić trzy bloki tematyczne. Pierwszy jest przeznaczony dla początkujących oraz dla potencjalnych użytkowników Amigi. Zawarte są w nim podstawowe informacje o komputerze (sposób przygotowania Amigi do pracy, podłączenia do telewizora lub monitora monochromatycznego) a także bardzo szczegółowe informacje na temat grafiki, dźwięku, systemu operacyjnego i pamięci. Specjalistyczna wiedza podana jest w sposób bardzo przystępny i bynajmniej nie ogranicza się do suchego przedstawienia faktów. Początkujący użytkownik znajdzie tutaj odpowiedź na wiele pytań, np. co to jest INTERLACE lub HAM, czym różni się pamięć CHIP od FAST itp.

Przyszłym użytkownikom Amigi polecam rozdział zatytułowany „Klan", w którym przedstawiona została prawie cała rodzina Amig (z wyjątkiem A600, bowiem książka była przygotowana do druku przed pojawieniem się tego komputera na rynku). W połączeniu z rozdziałem poświęconym peryferiom, w którym przedstawiono możliwości rozbudowy poszczególnych modeli Amig o dyski twarde, karty przyśpieszające zwane popularnie „dopalaczami" i karty grafiki 24-bitowej, pozwoli on wyrobić sobie szerszy pogląd na temat możliwości Amigi. Ten dział kończy krótki kodeks postępowania z komputerem, którego przestrzeganie pozwoli uniknąć kłopotliwej i kosztownej jego naprawy.

Druga część książki stanowi „prowadzący za rączkę" przewodnik po systemie operacyjnym. Początkujący amigowiec znajdzie tu wskazówki jak rozpocząć pracę z Workbenchem, jak formatować dyskietki, kopiować pliki itd. Ponadto znajdują się tu opisy najczęściej wykorzystywanych programów z dyskietki systemowej, a mianowicie Preferences i Info. Szkoda tylko, że we fragmencie opisującym preferencje zabrakło informacji o ustawieniu parametrów drukarki.

Po przebrnięciu przez podstawy czytelnik ma do dyspozycji dużą porcję materiału dotyczącego AmigaDOS. Można tu znaleźć informacje o standardowych urządzeniach czy katalogach systemowych. Nie zabrakło oczywiście zestawienia wszystkich poleceń AmigaDOS (V1.3) wraz z przykładami zastosowań.

Na końcu tej części autorzy umieścili prawdziwy rodzynek: opis sposobów wykorzystania niektórych procedur systemowych z poziomu asemblera - bardzo przydatne dla wszystkich, którzy chcą pisać własne programy w tym języku.

Trzecia część książki to różnego rodzaju dodatki. Na początku zamieszczony został leksykon terminów związanych z Amigą. Można w nim znaleźć odpowiedź na pytania: co to jest Frame Buffer, DMA, Genlock i wiele innych. W tej części zamieszczono również krótką instrukcję najpopularniejszego edytora tekstów, a mianowicie CygnusEd Professional. Nie zabrakło oczywiście objaśnienia kodów błędów, i to zarówno AmigaDOS-u, jak i Guru. Natomiast ci, którzy pragną skorzystać z procedur systemowych, znajdą tu ich pełny wykaz.

Podsumowując: książka „Po prostu Amiga" jest warta polecenia zwłaszcza dla początkujących użytkowników, chociażby ze względu na to, że prowadząc „za rączkę" pozwala zgłębić tajemnice Amigi. Bardziej zaawansowani użytkownicy również znajdą tu bardzo wiele informacji, które będą im służyły podczas poważniejszych „zmagań" z Amigą.

Szczególnie polecam tę książkę osobom pragnącym w niedalekiej przyszłości nabyć Amigę. Pozwoli ona zdecydować się na konkretny model, a później będzie służyła jako przewodnik po tajemniczym świecie Amigi.

PAWEŁ GALAS

Andrzej Bobek, Bartosz Smaga:
„Po prostu Amiga", str. 162
Wydawca: Wydawnictwo SOETO, Warszawa, 1992

COMMODORE BASIC I JEGO ODMIANY

Recenzję tej książki zacznę od jej wizytówki i czegoś, co powinno być jej pierwszą reklamówką - tytułu. „Commodore BASIC i jego odmiany" - brzmi ciekawie. Wiemy, że C-64 ma BASIC V2.0, seria C-16/C-116/+4 - BASIC V3.5, a poczciwe C-128 - aż V7.0. Nie wiadomo dokładnie, co stało się z wersjami pośrednimi (bo tak na moją głowę, to pomiędzy 3.5 a 7.0 zmieścić się mogą co najmiej 3 wersje). Być może pokutują w jakichś starszych komputerach z literką C i dwiema chorągiewkami (kłaniają się PET'y i cała menażeria komputerów typowo biurowych, wykopalisk z monitorem, klawiaturą i stacją dysków w jednej obudowie). Pamiętamy też przeróżne rozszerzenia BASIC V2.0 spowodowane głównie jego żałosną wprost ubogością w dziedzinach grafiki i dźwięku oraz niesamowitą elastycznością, o której tacy np. atarianie mogą jedynie marzyć. Wymienić tu starczy kilka ciekawych prób - SIMON'S BASIC, SUPER EXPANDER, WARSAW BASIC. Oprócz tych gigantów istnieją też setki i tysiące programików króciutkich, napisanych dla doraźnej potrzeby, dla wprawki lub dla żartu (cel dobry jak każdy inny, a bo co?).

W sumie temat - rzeka, na cztery doktoraty i jedną profesurę (nadzwyczajną). Bo, bądźmy szczerzy,, komuż - ach! - komuż, poza nieliczną kastą zapaleńców tudzież archeologów, chciałoby się o tym czytać. I to całą książkę!

Tytuł jednak kłamie. Po prostu i zwyczajnie. Bowiem treścią tej książki nie jest żadne syntetyczne opracowanie odmian języka BASIC, najczęściej mieszanego z błotem, ale także najczęściej używanego.

„Commodore BASIC i jego odmiany" składa się z dwóch części. W pierwszej, zajmującej 20 stron, ok. 1/7 całej książki, Autorzy skrótowo i po łebkach opisują BASIC V2.0, czyli wredne ścierwo zaimplementowane we wszystkich C-64, nawet w tych z płaską obudową. Część druga, znacznie dłuższa, to po prostu spolszczona instrukcja obsługi jednego z rozszerzeń BASIC, znanego powszechnie jako SIMON'S BASIC. Nie ma żadnej obietnicy kontynuacji, ni znaku, że książka ta jest elementem większej całości, opisującej inne odmiany i rozszerzenia BASIC. NIC.

Czyli tytuł nie jest dobrany do treści. Zamiast wizytówką staje się kamuflażem treści książki. Wniosek na przyszłość - nie kupuj książek nie zapoznawszy się najpierw z ich treścią. Inaczej możesz się dość przykro naciąć.

Zaś sama książka sprawia wrażenie, jakby musiała być gotowa na przedwczoraj. Duża rzutkość, prędkość i operatywność to cechy chwalebne u oficyny wydawniczej. Źle się jednak dzieje, gdy pośpiech idzie w parze z niestarannością. Naprawdę, warto by opóźnić wydanie tej książki o choćby tydzień tylko po to, by wyłapać w niej błędy, nieścisłości i pomyłki. Jeśli książka ma mieć walory dydaktyczne i poznawcze (a „Commodore BASIC..." rości sobie do tego pretensje), trzeba postarać się o usunięcie z niej wszystkich błędów przed oddaniem jej do druku. Nie zaszkodziłoby wpisanie i przetestowanie wszystkich listingów. W książce po prostu roi się od błędów w listingach! Zadałem sobie trud sprawdzenia kilku z nich. W niektórych błędy są oczywiste, inne zaś programy chyba na zawsze pozostaną dla mnie zagadką. Po prostu nie jestem na tyle biegły w SIMON'S BASIC-u, by je poprawić.

Ale nie będę gołosłowny: strona 56. Są tu dwa listingi. Pierwszy ma trzy linie, z czego jedną błędną, drugi - dwie, obie dla interpretera po prostu nie do przyjęcia. Strona 55: trzy listingi - jeden błąd. Strona 48: jeden listing - jeden błąd. Strona 114: listing wpisywany najwyraźniej z głowy, jego działanie nie jest zgodne z przewidywaniami Autora (a może znowu korrrekta!). Przykłady te można by, niestety, mnożyć - ale po co?

W listingach jest też kilka niekonsekwencji, które nie stanowią problemu dla mnie (he he!), starego wyjadacza. Początkującym mogą jednak zdrowo napsuć krwi. Na przykład - na początku Autorzy solennie obiecują, że wszystkie klawisze specjalne (RETURN, CLR/HOME etc.) opisywane będą tłustym drukiem a do tego kursywą. Deklaracja taka jest na stronie 39. Ale już na stronie 41 Autorzy wyłamują się z konwencji, którą sami wymyślili. W listingach nagminnie piszą: PRINT „SHIFT CLR/HOME". Może to i jest śmieszne, ale ja NAPRAWDĘ widziałam ludzi, którzy przepisali wstukując po kolei s, h, i, f, t itd. a potem dziwili się, że ekran nie jest czyszczony. A nie prościej byłoby napisać PRINT CHR$(147)? Nie?

Na zakończenie wypadałoby napisać o książce kilka ciepłych słów, ale nie wiem czy mi wyjdzie. Po lekturze doszedłem do wniosku, że jest przeznaczona dla ludzi, którzy kupili u pirata nielegalną kopię „sajmona", przegrali go od kolegi albo weszli w posiadanie jego kopii w podobny sposób, a teraz nie wiedzą co z nim począć. Przyda się też tym, którzy kupili „sajmona" za granicą i „w nagrodę" dostali instrukcję obsługi po duńsku albo w suahili. Natomiast tym, którzy „sajmona" nie mają i mieć nie chcą - książkę zdecydowanie odradzam. Bo jest ona po prostu instrukcją obsługi SIMON'S BASIC-a - ot i wszystko!

BARTŁOMIEJ KACHNIARZ

„Commodore BASIC i jego odmiany", str. 143
Pod redakcją Zbigniewa Jasiuka
Wydawca: Agencja Wydawnicza „M&M", Warszawa 1992

GRAFIKA DLA KAŻDEGO
CZ. 2

Zgodnie z zapowiedzią z poprzedniego odcinka dzisiaj przedstawiam listę wszystkich poleceń BASIC'a dotyczących grafiki. Ze względu na objętość tego materiału muszę się ograniczyć tylko do poleceń wspólnych dla omawianych komputerów (C-16/116/+4/128), a więc pomijam znaczną grupę poleceń BASIC'a V7.0 dotyczącą „duszków" (sprites). W przypadku kilku poleceń występują nieznaczne różnice składni w interpreterze języka BASIC V3.5 i V7.0 - zaznaczyłem to bezpośrednio w opisie.

Wyjaśnienia wymaga także ogólny sposób zapisu składni poleceń. Jako pierwsze występuje zawsze słowo kluczowe, dzięki któremu komputer dowiaduje się, co będzie miał za chwilę zrobić, a zaraz po nim podajemy kilka liczb oddzielonych przecinkami, które już dokładnie określają postawione przez nas zadanie. Liczby te mogą określać numer trybu graficznego, który chcemy włączyć, promień okręgu, współrzędne rogu prostokąta, itp. Czasami nie potrzebujemy podawać jakiegoś parametru i wtedy po prostu go pomijamy, a komputer sam wpisze sobie w to miejsce z góry umówioną wartość. Dla przykładu: gdy chcemy narysować zamalowany kwadrat, piszemy: BOX ,10,10,30,30,,1 . Spójrz na opis polecenia BOX (poniżej). Pominęliśmy pierwszy parametr - kolor, jakim ma być wykonany rysunek (i komputer przyjął, że ma to być kolor znaków) oraz parametr drugi od końca - kąt obrotu figury (zgodnie z opisem komputer przyjął go jako 0 stopni i kwadrat nie jest przekręcony).

Jeżeli w wyszczególnionych poleceniach któryś parametr będzie można pominąć, to jego nazwę umieszczę w nawiasie kwadratowym, a pod spodem wyjaśnię, jaką liczbę wstawi komputer na jego miejsce w przypadku jego pominięcia. Nawiasy kwadratowe są więc tylko informacją dla Czytelników i nie należy ich wpisywać do komputera!

Współrzędne punktów, które podajemy jako parametry poleceń, oznaczam jako x1,y1, x2,y2, itd. Numery przy literach mówią, którego punktu jest to współrzędna, np. x3 oznaczałoby współrzędną X trzeciego punktu. Jeżeli nie bardzo rozumiesz, co to są współrzędne punktów, to musisz zdobyć poprzedni numer „C&A" - tam to wyjaśniałem.

Przed przystąpieniem do właściwego tematu wyjaśniam od razu znaczenie parametru, który będzie się powtarzał w kilku poleceniach:
kolor - mówi komputerowi, jakim kolorem ma być wykonywany rysunek:
0 - kolorem z rejestru tła (co oznacza po prostu wycieranie),
1 - kolorem z rejestru pierwszego planu (znaków),
2 - kolorem z rejestru „multicolor 1",
3 - kolorem z rejestru „multicolor 2" (dwie ostatnie liczby można podstawiać w miejsce parametru tylko wtedy, gdy rysujemy w trybie graficznym 3 i 4).

Najczęściej pomijamy ten parametr - komputer przyjmuje wtedy jako wartość domyślną liczbę 1, czyli kolor pierwszego planu. Do przyporządkowania rejestrom konkretnych kolorów używamy polecenia COLOR.

Często używany skrót PC (kursor graficzny) objaśniłem przy opisie polecenia LOCATE. Radzę zajrzeć tam na początku.

Spis komend BASIC'a V3.5/7.0 dotyczących grafiki.

BOX [kolor],x1,y1,[x2,y2],[obrót][,zam]
Polecenie to służy do rysowania prostokątów (kwadratów).
x1,y1 - współrzędne lewego, górnego rogu prostokąta
x2,y2 - współrzędne prawego, dolnego rogu prostokąta (standardowo współrzędne PC)
obrót - kąt obrotu prostokąta względem jego środka podany w stopniach (standardowo 0)
zam - czy prostokąt ma być zamalowany: 0 - nie, 1 - tak (standardowo 0)
Przykłady:
BOX ,30,40,120,90 - rysuje prostokąt
BOX ,120,50,150,80,45 - rysuje obrócony kwadrat
BOX ,40,20,,60,1 - rysuje obrócony, zamalowany prostokąt z punktu (40,20) do aktualnego położenia PC

CHAR [kolor],x,y,[łańcuch][,rvs]
Umożliwia wyświetlanie na ekranie zwykłego tekstu (liter).
x - kolumna (0-39), od której ma być wyświetlany napis
y - wiersz (0-24), w którym ma być umieszczony napis
łańcuch - tekst umieszczony w cudzysłowach lub zmienna tekstowa (w przypadku pominięcia parametru nic nie jest wyświetlane)
rvs - czy tekst ma być wyświetlony w negatywie: 0 - nie, 1 - tak (standardowo 0)
Przykład:
CHAR,2,2,"NAPIS W GRAFICE",1 - wyświetla napis w odwróconych barwach poczynając od współrzędnych tekstowych (2,2), czyli od punktu (16,16) we współrzędnych graficznych (bo na jeden znak przypada 8 punktów)
Uwaga: w grafice wielokolorowej ze względu na mniejszą rozdzielczość standardowe znaki będą miały nieco zniekształcony kształt.

CIRCLE [kolor],[x1,y1],xr,[yr],[kąt1],[kąt2],[obrót],[segment]
Jest to polecenie uniwersalne - da się nim narysować niemal każdą figurę, lecz stosujemy je przede wszystkim do rysowania okręgów, elips, wielokątów, i ich wycinków.
x1,y1 - współrzędne środka (standardowo współrzędne PC)
xr - długość promienia wzdłuż osi X
yr - długość promienia wzdłuż osi Y (w przypadku pominięcia parametru yr=xr, a więc promienie są równe i rysowany jest okrąg)
kąt1 - kąt (podany w stopniach), od którego komputer rozpoczyna rysowanie figury (standardowo 0)
kąt2 - kąt końca wycinka rysowanej przez komputer figury (standardowo 360; znaczenie parametrów kąt1 i kąt2 wyjaśnia rys. 1)
obrót - kąt obrotu figury względem jej środka, podany w stopniach (standardowo 0)
segment - kąt między rysowanymi segmentami figury (standardowo 2)
Ten parametr wymaga dłuższego komentarza. Nasze komputery zamiast rysować okrąg rysują wielokąt wpisany w ten okrąg (patrz rys. 2), i nie ma w tym nic złego, gdyż przy dużej ilości boków wielokąt pokrywa się z wymaganym okręgiem. Przy okazji możemy narysować dowolny, niekoniecznie „przylegający" do okręgu wielokąt, wpisując w miejsce parametru „segment" wyrażenie 360/n, gdzie n jest liczbą boków wielokąta (patrz rys. 3).
Przykłady:
CIRCLE ,160,100,50 - rysuje okrąg na środku ekranu
CIRCLE ,,50,30 - rysuje elipsę ze środkiem w punkcie zawierającym PC
CIRCLE ,100,80,40,,90,270 - rysuje dolną połowę okręgu
CIRCLE ,60,40,20,18,,,,45 - rysuje ośmiokąt
Uwaga: w prosty sposób i bez wyraźnej utraty jakości możemy przyspieszyć rysowanie okręgów: wystarczy jako parametr „segment" podawać kąt 12. Dzięki temu polecenie CIRCLE,160,100,50,,,,,12 jest wykonywane około pięciu razy szybciej od CIRCLE ,160,100,50 , a utrata jakości jest niezauważalna.

COLOR rejestr,kolor [,jasność] (C-16)
COLOR rejestr,kolor (C-128)
Umożliwia wpisanie numerów kolorów do rejestrów (są to specjalne komórki pamięci, z których informacja o kolorze jest pobierana podczas wyświetlania obrazu).
Rejestr:
0 - tło
1 - pierwszy plan (znaki lub rysunek)
2 - multicolor 1
3 - multicolor 2
4 - ramka
kolor - liczba z zakresu 1 - 16 określająca kolor zgodnie z tabelami z instrukcji obsługi
jasność (tylko C-16) - liczba od 0 (najciemniej) do 7 (najjaśniej). Standardowo - 7.
Przykłady:
C-128: COLOR 1,8 - od tej pory rysunki będą wykonywane w kolorze żółtym
C-16: COLOR 1,8,2 - ustawia kolor pierwszego planu na ciemny odcień koloru żółtego i komputer będzie rysować tym kolorem

DRAW [kolor][,x1,y1,] [,x2,y2]... [TO x3,y3] [TO x4,y4]...
Polecenie służy do rysowania punktów, odcinków i dowolnych łamanych.
x1,y1 - współrzędne rysowanego punktu lub początku kreski (standardowo pozycja PC)
x2,y2,x3,y3,x4,y4,... - współrzędne kolejnych punktów
Przykłady:
DRAW 1,10,10 - rysuje punkt
DRAW ,20,40,30,20,40,30 - rysuje trzy punkty
DRAW ,50,30 TO 100,90 - rysuje kreskę
DRAW 1 TO 160,130 - rysuje kreskę od aktualnej pozycji PC do punktu (160,130)
DRAW TO 50,25 TO 40,30 TO 60,40 - rysuje łamaną złożoną z trzech odcinków (początek pierwszej kreski w PC)

GRAPHIC tryb [,czyść] lub GRAPHIC CLR (C-16)
GRAPHIC tryb [,czyść] [,podział] lub GRAPHIC CLR (C-128)
Polecenie włącza określony parametrami tryb graficzny.
tryb:
0 - tekst (40 kolumn)
1 - grafika wysokiej rozdzielczości (hires)
2 - grafika wysokiej rozdzielczości, dolna część ekranu przeznaczona na wyświetlanie tekstu
3 - grafika wielobarwna (multicolor)
4 - grafika wielobarwna, dolna część ekranu przeznaczona na wyświetlanie tekstu
5 (tylko C-128) - tekst 80 kolumn
czyść: czy ekran ma być czyszczony przy włączaniu trybu:
0 - nie, 1 - tak (standardowo 0)
podział: (tylko C-128) w trybach graficznych 2 i 4 oznacza numer linii, od której ma być wyświetlany tekst (standardowo 19)
Polecenie GRAPHIC CLR powoduje powrót do wyświetlania tekstu oraz zwalnia pamięć obrazu graficznego (ekran graficzny zajmuje w pamięci aż 12 KB).

Przykłady:
GRAPHIC 1,1 - włącza tryb graficzny wysokiej rozdzielczości i czyści ekran
GRAPHIC 2,0,5 (tylko C-128) - przeznacza 5 pierwszych linii na wyświetlanie obrazu wysokiej rozdzielczości, a resztę ekranu na wyświetlanie tekstu, nie czyści ekranu

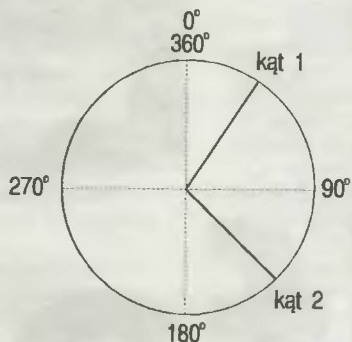
LOCATE x1,y1
Ustawia kursor graficzny (ang. Pixel Cursor, w skrócie PC) w żądanym miejscu ekranu.
x1,y1 - współrzędne ekranowe punktu, na którym ma być ustawiony PC
Uwaga: kursor graficzny jest niewidoczny i ma wielkość jednego punktu graficznego. Znajduje się on zawsze w ostatnio rysowanym punkcie, chyba że przestawimy go gdzie indziej poleceniem LOCATE. Pozycję PC można odczytać za pomocą funkcji RDOT. Dzięki kursorowi graficznemu w wielu przypadkach można uprościć wykonywanie rysunków - do tego tematu powrócę za miesiąc.

PAINT [kolor][,x1,y1][,tryb]
Zamalowuje ograniczony obszar podanym kolorem.
x1,y1 - punkt, od którego zostanie rozpoczęte zamalowywanie
tryb:
0 - zamalowywany obszar jest ograniczony kolorem źródłowym (podanym w poleceniu). Wartość ta jest przyjmowana jako standardowa, jeżeli pominiemy parametr wydając polecenie.
1 - zamalowany obszar jest ograniczony dowolnym kolorem różnym od koloru tła. W praktyce podawanie parametru „tryb" ma sens tylko przy stosowaniu trybu wielobarwnego.
Uwagi:
1) jeżeli obszar nie jest ograniczony, to może zostać zamalowany cały ekran;
2) punkt, który podajemy jako parametr, musi mieć kolor inny od tego, którym chcemy zamalowywać, w przeciwnym wypadku polecenie nie da żadnego efektu.
Przykład:
CIRCLE ,100,100,50:PAINT ,100,100

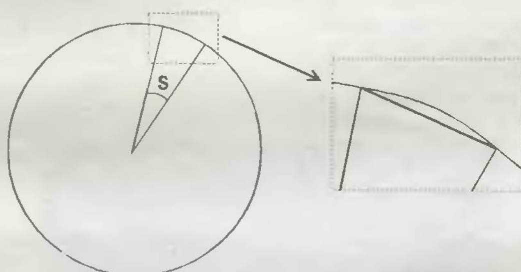
SCALE n (C-16)
SCALE n [,xmax,ymax] (C-128)
Polecenie pozwala zmienić skalę wykonywanego rysunku. Normalnie ekran ma wymiary 320 na 200 (hires) lub 160 na 200 punktów (multicolor). SCALE pozwala nam zmienić te wymiary, jednak rzeczywista rozdzielczość się nie zmieni - po prostu, gdy będzie to konieczne, kilka podanych punktów będzie przyporządkowanych jednemu, fizycznemu punktowi ekranowemu. Polecenie ma niewielkie zastosowanie praktyczne.
n: 1 - skalowanie włączone, 0 - skalowanie wyłączone
xmax - liczba z zakresu 320-32767 (standardowo 1023)
ymax - liczba z zakresu 200-32767 (standardowo 1023)
Wykonanie polecenia GRAPHIC wyłącza skalowanie.
Przykład:
10 GRAPHIC 1,1
20 SCALE 1
30 BOX ,0,0,320,200
Proszę zwrócić uwagę, że prostokąt ma wymiary standardowego ekranu, a dzięki skalowaniu zajął tylko niewielką jego część.

SCNCLR
Czyści ekran i umieszcza PC w punkcie (0,0) - lewy, górny róg ekranu.

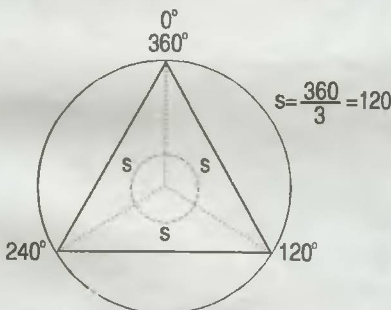
SSHAPE nazwa,x1,y1[,x2,y2]
Zapamiętuje podany obszar ekranu graficznego w zmiennej BASIC'a.
nazwa - nazwa zmiennej tekstowej, w której będzie zapamiętany obszar
x1,y1 - współrzędne lewego, górnego rogu zapamiętywanego obszaru
x2,y2 - współrzędne prawego, dolnego rogu zapamiętywanego obszaru (standardowo pozycja PC)
Ze względu na fakt, że zmienna tekstowa może zawierać co najwyżej 255 znaków, wielkość zapamiętywanego obszaru także jest ograniczona. Instrukcje obsługi podają wzór na obliczanie maksymalnego obszaru, który można w ten sposób zapamiętać.



Rysowanie fragmentu okręgu za pomocą polecenia CIRCLE.



s - kąt określony przez parametr SEGMENT polecenia CIRCLE.



Zasada rysowania wielokątów poleceniem CIRCLE z zastosowaniem parametru SEGMENT (na rysunku -s)

GSHAPE nazwa,[x1,y1][,tryb]
Odtworzenie obszaru zapamiętanego w zmiennej za pomocą polecenia SSHAPE.
nazwa - nazwa zmiennej tekstowej, w której zapamiętano obszar
x1,y1 - współrzędna lewego, górnego rogu, od którego zostanie narysowany zapamiętany obszar
tryb:
0 - odtworzenie obszaru tak samo, jak został pobrany
1 - odtworzenie obszaru w negatywie
2 - nałożenie odtwarzanego obszaru na obraz już znajdujący się w tym miejscu na ekranie (logiczne LUB)
3 - wyświetlenie tylko tych części odtwarzanego obszaru, które pokrywają się z rysunkiem już znajdującym się (logiczne I)
4 - wykonuje na obszarze odtwarzanym i znajdującym się pod nim rysunkiem logiczną operację ALBO
Przykład: w zmiennej K$ zostaje zapamiętany rysunek zamalowanego okręgu, który jest następnie odtworzony we wszystkich możliwych trybach.

```
10 GRAPHIC 1,1
20 CIRCLE ,8,8,7: PAINT ,8,8
30 SSHAPE K$,0,0,16,16: REM ZAPAMIETANIE OBSZARU
40 BOX,40,80,150,116,,1: CHAR,7,11,"TRYB:",1
50 GSHAPE K$,50,110: CHAR,6,12,"0",1
60 GSHAPE K$,70,110,1: CHAR,9,12,"1",1
70 GSHAPE K$,90,110,2: CHAR,12,12,"2",1
80 GSHAPE K$,110,110,3: CHAR,15,12,"3",1
90 GSHAPE K$,130,110,4: CHAR,17,12,"4",1
```

RCLR (n)
Polecenie umożliwiające odczytanie aktualnego koloru umieszczonego w rejestrach kolorów.
n:
0 - tła trybu tekstowego (40 kolumn)
1 - pierwszego planu (znaki i rysunki)
2 - multicolor 1
3 - multicolor 2
4 - ramki obrazu
5 (tylko C-128) - znaków trybu tekstowego (40 lub 80 kolumn)
6 (tylko C-128) - tło trybu znakowego 80 kolumn
Przykłady:
PRINT RCLR (4) - wyświetla numer odpowiadający kolorowi ramki
K=RCLR (0) - podstawia numer koloru tła pod zmienną K

RDOT (n)
Polecenie umożliwiające odczytanie aktualnej pozycji PC lub koloru punktu o współrzędnych PC.
n:
0 - zwraca współrzędną X kursora graficznego
1- zwraca współrzędną Y kursora graficznego
2- zwraca numer źródła koloru spod kursora graficznego (0 - 3).
Innymi słowy parametr ten jest stosowany do testowania, czy punkt jest zapalony, czy zgaszony (niezbędne w wielu grach).
Przykład:
PRINT „KURSOR GRAFICZNY ZNAJDUJE SIE W PUNKCIE"; RDOT(0); RDOT(1)

RGR (x)
Zwraca numer aktualnego trybu graficznego.
x - dowolna liczba (musi wystąpić)
Przykład:
PRINT RGR(1)

RLUM (N) (C-16)
Zwraca numer odcienia (jasności) koloru z rejestru określonego parametrem n.
n:
0 - tło trybu tekstowego (40 kolumn)
1 - pierwszy plan (znaki i rysunki)
2 - multicolor 1
3 - multicolor 2
4 - ramka obrazu
Przykład:
PRINT RLUM (1) - wyświetla numer jasności koloru znaków

I to już cała potrzebna nam teoria, w następnym odcinku powrócimy do zabawy, a przez ten czas radzę dobrze zapoznać się ze składnią wszystkich poleceń i przećwiczyć wszystkie przykłady.

WOJCIECH KAZIMIERCZAK

LIFE is life

Gra w życie znana jest od bardzo dawna. Jej ideą jest obserwacja rozwoju kolonii „bakterii". Na początku zakładamy zalążek koloni, a następnie obserwujemy jak „komórki" rozmnażają się i umierają, tworząc przy okazji ciekawe wzory. Dzisiaj chcielibyśmy przedstawić pewną mutację tej gry.

W pierwszym etapie, przy pomocy joysticka ustawiamy kolonie „bakterii", klawiszami F1-F7 zmieniamy ich „gatunek" (różne gatunki reprezentowane są przez różne kolory). Kiedy wszystko ustawimy tak jak chcemy, klawiszem RETURN kończymy etap edycji i przechodzimy do następnego, w którym nasza rola ogranicza się jedynie do obserwacji.

Bakterie „żyją", ale w odróżnieniu od pierwowzoru, ich rozwój nie przebiega według określonych zasad, lecz losowo. Odbywa się to w ten sposób, że komputer sam wybiera daną komórkę i w zależności od tego, jakiego jest gatunku, zapala cztery sąsiednie (przylegające bokami), w odpowiednim kolorze. W każdej chwili mamy możliwość wciśnięcia klawisza (strzałka w lewo), co spowoduje, iż gra zostanie przerwana, i przejdziemy do trybu edycji. Teraz możemy zmodyfikować „pole walki", a następnie wrócić do gry klawiszem RETURN.

Kiedy znudzi nam się już zabawa z daną kolonią bakterii , możemy wcisnąć klawisz RESTORE i zacząć wszystko od nowa. Na początku jest zadawane pytanie, czy kolor 0 (czarny) ma być neutralny. Jeżeli odpowiemy, że nie, to kolor ten będzie traktowany jak szczep bakterii. Natomiast odpowiedź twierdząca oznacza, że będzie traktowany jako neutralne tło.

Życzę udanej zabawy.

RAFAŁ PIASEK

```
100 rem **************************
101 rem *coded by jetboy/cavern*
102 rem **************************
103 rem
104 d=49152:b= 197802
105 c=0:e=d
106 read a$:ifa$="end" then 116
107 a1=asc(left$(a$,1)) and 63
108 a2=asc(right$(a$,1)) and 63
109 ifa1>47then111
110 a1=a1+9:goto112
111 a1=a1-48
112 ifa2>47thena2=a2-48:goto114
113 a2=a2+9
114 a=a1*16+a2:poked,a
115 d=d+1:c=c+a:goto106
116 ifc<>bthenprint"blad w
    liniach data":stop
117 syse
118 data 78,a2,0f,a0,00,b9,1a,c0
119 data 99,01,08,c8,d0,f7,ee,07
120 data c0,ee,0a,c0,ca,d0,ec,4c
121 data 0b,08,0b,08,50,06,9e,32
122 data 30,35,31,00,a0,00,a2,1b
123 data 78,e6,01,e0,06,b0,05,bd
124 data fd,0c,95,fa,bd,03,0d,95
125 data a0,ca,10,ef,b0,1f,0d,99
126 data e8,06,c8,d0,f7,4c,07,07
127 data cd,bd,1a,1d,2e,d4,8e,f1
128 data a0,50,27,10,0c,88,d0,7b
129 data f6,e7,0a,68,08,ff,f0,7b
130 data f4,d8,aa,9a,4c,00,10,a9
131 data 3b,8d,11,ff,e0,10,09,10
132 data 8d,16,d0,ad,3f,20,09,08
133 data 8d,18,7f,02,8e,20,d0,8e
134 data 21,d0,fd,51,0f,55,d8,01
135 data 40,d9,50,da,12,db,a9,bc
136 data 07,40,04,50,05,14,05,9d
137 data 00,07,07,e8,d0,e1,a0,20
138 data 20,2d,ff,13,20,ed,60,f9
139 data 82,8a,15,07,8e,37,21,35
140 data 43,de,1d,0a,8c,e8,29,26
141 data 48,49,03,70,10,8f,68,ae
142 data 45,ba,e3,46,3f,a6,a2,ff
143 data ec,12,d0,d0,fb,a9,ff,fe
144 data ec,84,c9,3d,ef,f0,39,c9
145 data df,f0,3d,c9,ff,bf,f0,41
146 data c9,f7,f0,45,c9,ff,fd,f0
147 data 23,ff,02,bf,85,1e,a8,32
148 data 38,0a,29,08,f0,cf,69,90
149 data 04,f0,54,39,01,f9,03,2f
150 data 90,02,f0,3a,98,29,10,3f
151 data f0,65,98,4c,f6,46,34,01
152 data 73,c1,1e,02,f0,47,03,bc
153 data 01,a9,00,8d,ef,91,1a,97
154 data 82,f8,4f,78,47,2f,62,41
155 data f0,9e,34,4c,4c,7e,06,ca
156 data e0,ff,fe,00,0b,e0,42,11
157 data ae,6f,20,b0,40,d0,02,e7
158 data a2,3f,8e,f4,4c,3d,11,ad
159 data 0f,43,b0,2a,ad,44,12,47
160 data 94,ad,42,36,68,dc,4c,52
161 data 11,a9,ff,8d,0e,7f,d5,8d
162 data 0f,d4,a9,81,8d,12,ff,d4
163 data a8,47,30,aa,ea,ea,ad,1b
164 data bf,d4,29,3f,18,69,43,a8
165 data 8e,ff,b0,8c,6c,20,fc,13
166 data 8d,1e,e0,f0,00,cc,03,c8
167 data d3,0c,88,f4,c1,e8,bd,30
168 data ad,3f,12,ae,40,5f,12,ac
169 data 41,12,ca,20,47,12,ff,a9
170 data 7f,8d,00,dc,ad,01,dc,ff
171 data 29,02,d0,a1,20,d4,10,4c
172 data ff,da,11,d0,62,9e,29,03
173 data 84,ff,a8,b9,fd,b1,8d,5c
174 data 12,b9,9e,70,8d,5b,12,a4
175 data cf,ff,4c,ff,ff,f7,80,15
176 data fd,50,1e,cc,13,f9,1d,b3
177 data e1,83,36,15,19,7d,6c,8f
178 data 19,5b,10,3e,18,cf,66,10
179 data 17,7d,ba,1d,52,1a,67,58
180 data 5f,e6,16,ff,11,07,ea,87
181 data 40,11,87,11,fd,91,fd,ef
182 data 60,a9,5d,8d,f2,f9,8e,8d
183 data 03,f3,90,c5,8d,f4,39,fc
184 data 8d,03,15,90,f6,1a,40,f7
185 data 69,01,f8,14,a9,12,8d,f9
186 data bf,14,fe,3c,17,03,bc,12
187 data 9d,03,0d,a9,1f,3d,3f,03
188 data 98,18,7d,ba,9d,03,2d,3a
189 data c8,d0,db,a2,05,1f,bd,90
190 data 29,1f,36,98,05,1d,a0,9d
191 data 95,13,38,e8,e0,03,65,d0
192 data ea,60,a2,00,8a,fe,80,20
193 data a0,21,28,22,0a,23,02,80
194 data 24,a0,25,28,26,0a,27,02
195 data 80,28,a0,29,28,2a,0a,2b
196 data 02,80,2c,a0,2d,28,2e,0a
197 data 2f,02,80,30,a0,31,28,32
198 data 0a,33,02,80,34,a0,35,28
199 data 36,0a,37,02,80,38,a0,39
200 data 28,3a,0a,3b,02,80,3c,a0
201 data 3d,28,3e,9d,00,3f,0f,e8
202 data d0,9d,60,8a,0a,aa,b0,ff
203 data 18,f4,07,c2,15,be,10,8f
204 data b9,fa,14,85,fd,b9,bf,8a
205 data 16,85,fe,8a,29,f8,a8,ff
206 data bd,52,18,31,fd,aa,bd,31
207 data ff,14,60,c4,cd,a0,6e,83
208 data 62,1b,10,d3,ef,10,af,bc
209 data 02,5e,90,0c,7b,3f,f3,02
210 data e8,3e,07,00,03,d8,02,7b
211 data a8,e3,40,41,42,43,ef,44
212 data 45,46,47,80,81,82,83,ff
213 data 84,85,86,87,c0,c1,c2,c3
214 data ff,c4,c5,c6,c7,00,01,02
215 data 03,ff,04,05,06,07,f2,86
216 data a7,bf,18,03,f0,04,25,7b
217 data f8.18,1f,80,23,2a,df,c0
218 data 18,fc,01,1e,2f,fe,18,07
219 data e0,08,34,f7,f0,18,3f,00
220 data 47,39,bf,81,18,f8,02,3e
221 data 3d,00,8f,21,40,23,22,d0
222 data 08,23,f4,02,24,3d,00,8f
223 data 26,40,23,27,d0,08,28,f4
224 data 02,29,3d,00,8f,2b,40,23
225 data 2c,d0,08,2d,f4,02,2e,3d
226 data 00,8f,30,40,23,31,d0,08
227 data 32,f4,02,33,3d,00,8f,35
228 data 40,23,36,d0,08,37,f4,02
229 data 38,3d,00,8f,3a,40,23,3b
230 data d0,08,3c,f4,02,3d,3d,00
231 data 8f,70,f7,3f,7d,3f,cf,cf
232 data f3,f3,fc,fc,ff,f8,c1,c0
233 data c0,30,f7,30,0c,0c,03,03
234 data ff,f8,07,c2,4c,10,10,04
235 data 04,01,01,ff,f8,83,80,80
236 data ee,08,b1,08,02,02,fc,41
237 data 29,0c,09,fb,06,f8,09,09
238 data 32,b3,03,0f,0d,0d,0f,7c
239 data ad,89,a2,26,20,b0,16,09
240 data 07,70,09,12,f8,31,39,39
241 data 32,cf,20,21,fc,2a,23,f7
242 data 43,19,42,c6,82,20,30,20
243 data b9,d3,06,25.03,20,0e,05
244 data 15,7d,c9,01,94,10,28,14
245 data 2f,b7,0e,29,20,3f,ff,81
246 data 5e,11,28,70,fb,d2,26,1f
```

```
247 data 6e,5d,0a,0f,19,be,d9,09
248 data 03,0b,3f,22,9a,0b,a1,02
249 data e3,b0,0f,60,fd,e4,06,31
250 data 2d,06,37,5f,ee,0f,17,4d
251 data a5,02,78,0c,57,f1,91,2e
252 data 34,a1,ef,72,14,15,1b,12
253 data 0e,fc,9c,18,1a,10,0f,03
254 data fc,c9,5c,22,6b,04,74,01
255 data 22,9f,21,79,40,d8,11,b1
256 data c8,91,14,01,12,97,14,ea
257 data 17,b3,0b,dc,1f,8b,22,3d
258 data 20,7b,1a,01,e3,1a,19,0d
259 data 01,1f,d8,a4,6c,0f,07,3a
260 data 0d,04,0f,91,12,01,d5,20
261 data 5d,5d,27,78,0d,10,12,7b
262 data 1a,05,0a,13,ff,05,c5,09
263 data 14,12,19,02,15,7d,02,12
264 data 05,13,0c,05,7c,01,31,5d
265 data 6b,fc,26,be,73,5d,17,03
266 data f8,0e,09,0a,87,20,04,c6
267 data 0f,0c,0e,1d,41,0c,01,17
268 data 09,13,1a,20,7f,02,19,20
269 data 0b,0f,0e,14,19,ff,0e,15
270 data 0f,17,01,03,5d,6d,ff,00
271 data 25,40,7d,fb,00,70,20,78
272 data 20,0d,e8,8d,10,0e,32,cd
273 data 2d,11,20,5b,ff,a9,87,fc
274 data ca,10,3f,19,03,27,58,b8
275 data 8d,a7,84,da,3c,1b,ae,23
276 data 24,f5,a9,ef,b7,70,8f,80
277 data 5c,50,16,fb,21,76,8c,7b
278 data 05,03,4c,58,9d,d3,2f,9e
279 data 04,a3,4c,81,5d,96,de,8d
280 data fb,11,3e,51,40,1c,08,7d
281 data b6,0c,1a,1e,40,80,00,80
282 data c0,80,00,00,00,00,01,00
283 data 00,04,09,08,06,07,08,07
284 data 07,0a,0b,0b,09,0c,0d,10
285 data 20,cd,07,91,fe,a9,ff,c7
286 data fe,d0,02,c6,ff,a5,fd,c9
287 data 07,d0,17,a5,fc,c9,e8,d0
288 data 11,c6,01,58,4c,00,10,a0
289 data 46,b1,fc,99,e8,07,88,10
290 data f8,c8,20,bb,07,b0,d1,a2
291 data 00,86,9b,86,9c,20,bb,07
292 data 90,3a,20,bb,07,90,31,20
293 data bb,07,90,28,20,bb,07,90
294 data 1f,20,bb,07,90,05,20,cd
295 data 07,d0,0f,a2,03,20,bb,07
296 data 26,97,ca,d0,f8,a5,97,18
297 data 69,06,85,97,a2,05,d0,10
298 data a2,05,d0,0a,a2,04,d0,06
299 data a2,03,d0,02,a2,02,86,97
300 data 20,bb,07,b0,06,a9,00,85
301 data 9e,f0,13,8a,69,03,aa,20
302 data bb,07,90,04,8a,69,03,aa
303 data b5,9a,85,9e,b5,a2,85,9f
304 data b5,ae,aa,20,bb,07,26,9b
305 data 26,9c,ca,d0,f6,38,a5,9b
306 data 65,fe,aa,a5,9c,65,ff,4a
307 data 8a,65,9e,85,9b,68,65,9f
308 data 85,9c,b1,9b,91,fe,a9,ff
309 data c7,9b,d0,02,c6,9c,c7,fe
310 data d0,02,c6,ff,c6,97,d0,ea
311 data 4c,f5,06,c6,fa,f0,03,46
312 data fb,60,a9,08,85,fa,20,cd
313 data 07,4a,85,fb,60,a9,ff,c7
314 data fc,d0,02,c6,fd,c6,01,8d
315 data 20,d0,8d,20,d0,8d,20,d0
316 data 8d,20,d0,e6,01,b1,fc,60
317 data 01,58,4c,00,10,a0,46,b1
318 data fc,99,e8,07,88,10,f8,c8
319 data 20,bb,07,b0,d1,a2,00,86
320 data 9b,86,9c,20,bb,07,90,3a
321 data 20,bb,07,90,31,20,bb,07
322 data 90,28,20,bb,07,90,1f,20
323 data bb,07,90,05,20,cd,07,d0
324 data 0f,a2,03,20,bb,07,26,97
325 data ca,d0,f8,a5,97,18,69,06
326 data 85,97,a2,05,d0,10,a2,05
327 data d0,0a,a2,04,d0,06,a2,03
328 data d0,02,a2,02,86,97,20,bb
329 data 07,b0,06,a9,00,85,9e,f0
330 data 13,8a,69,03,aa,20,bb,07
331 data 90,04,8a,69,03,aa,b5,9a
332 data 85,9e,b5,a2,85,9f,b5,ae
333 data aa,20,bb,07,26,9b,26,9c
334 data ca,d0,f6,38,a5,9b,65,fe
335 data aa,a5,9c,65,ff,48,8a,65
336 data 9e,85,9b,68,65,9f,85,9c
337 data b1,9b,91,fe,a9,ff,c7,9b
338 data d0,02,c6,9c,c7,fe,d0,02
339 data c6,ff,c6,97,d0,ea,4c,f5
340 data 06,c6,fa,f0,03,46,fb,60
341 data a9,08,85,fa,20,cd,07,4a
342 data 85,fb,60,a9,ff,c7,fc,d0
343 data 02,c6,fd,c6,01,8d,20,d0
344 data 8d,20,d0,8d,20,d0,8d,20
345 data d0,e6,01,b1,fc,60,00,00
    ,end
```

MEM VIEW V0.9

Jak ogólnie wiadomo pisanie programu nie jest rzeczą tak trudną, jak jego późniejsze uruchamianie. Trochę odwołań do nie istniejących adresów lub, co gorsza, skok do pamięci ROM stacji dysków i wywołanie procedury formatującej dyskietkę mogą doprowadzić do przysłowiowej szewskiej pasji. W takim przypadku warto wiedzieć, co nasze dzieło wyprawia z komputerem.

Jednym ze sposobów jest kontrolowanie wartości pewnych komórek pamięci RAM w trakcie działania programu. Do tego właśnie celu służy przedstawiony na listingu nr 1 program MEM VIEW. Po jego uruchomieniu komputer zapyta o adres dwóch komórek pamięci, których wartość chcemy podglądać. Po wpisaniu odpowiednich danych komputer wyświetli w lewym górnym rogu adresy wybranych komórek oraz ich wartości w kodzie szesnastkowym.

Program można zmodyfikować tak, aby pokazywał więcej niż dwie komórki, np. szesnaście lub nawet całą stronę pamięci. Można również dokonać takich zmian, aby w każdej chwili można było zmienić adres „podglądanej" komórki lub też jej wartość.

Dla wszystkich, którzy zainteresowani są modyfikacją programu, podajemy listing źródłowy (2) - TURBO ASSEMBLER V5.1 na C-64.

Powodzenia w przeróbkach życzy

MARIUSZ FERDYN

```
200 rem ************************
205 rem *    Mem View V.0.9    *
210 rem *          by          *
215 rem *      M. Ferdyn       *
220 rem ************************
225 d=49152:b=10294
230 c=0:e=d
235 read a$:if a$="end" then 285
240 a1=asc (left$ (a$,1)) and 63
245 a2=asc (right$ (a$,1))and 63
250 if a1>47 then 260
255 a1=a1+9:goto 265
260 a1=a1-48
265 if a2>47 then a2=a2-48:goto 275
270 a2=a2+9
275 a=a1*16+a2:poke d,a
280 d=d+1:c=c+a:goto 235
285 if c<>b then print "blad w liniach
    data":stop
290 print chr$ (147)
295 input "adres pierwszej komorki :";a
300 input "adres drugiej komorki   :";b
305 gosub 400
310 poke 251,mb:poke 252,sb
315 a=b:gosub 400
320 poke 253,mb:poke 254,sb
325 print:print "activ."
330 sys e:new
335 data ad,14,03,8d,34,03,ad,15
340 data 03,8d,35,03,78,a9,19,8d
345 data 14,03,a9,c0,8d,15,03,58
350 data 60,a0,00,a5,fc,20,4c,c0
355 data a5,fb,20,4c,c0,a2,00,a1
360 data fb,20,45,c0,a9,20,20,61
365 data c0,a5,fe,20,4c,c0,a5,fd
370 data 20,4c,c0,a2,00,a1,fd,20
375 data 45,c0,6c,34,03,48,a9,2d
380 data 20,61,c0,68,48,4a,4a,4a
385 data 4a,20,55,c0,68,29,0f,c9
390 data 0a,90,04,e9,09,d0,02,09
395 data 30,99,00,04,c8,60,end
400 sb=int (a/256):mb=a-(sb*256)
405 return
```

```
         *= $C000
;-------------------------
;   $FB, $FC - ADRES PIERWSZEJ KOMORKI,
;              ODPOWIEDNIO MLODSZY
;              STARSZY BAJT.
;
;   $FD, $FE - ADRES DRUGIEJ KOMORKI,
;              KTOREJ WARTOSC CHCEMY
;              PODGLADAC.
;-------------------------
START
          LDA $0314
          STA $0334
          LDA $0315
          STA $0335
          SEI
          LDA #<IRQ
          STA $0314
          LDA #>IRQ
          STA $0315
          CLI
          RTS
;-------------------------
IRQ
          LDY #$00      ;WYDRUK 1 ADRESU
          LDA $FC       ;ORAZ JEGO
          JSR PRT       ;ZAWARTOSCI
          LDA $FB
          JSR PRT
          LDC #$00
          LDA ($FB,X)
          JSR SPCPRT

          LDA #$20      ;WYDRUK 2 ADRESU
          JSR LB        ;ORAZ JEGO
                        ;ZAWARTOSCI
          LDA $FE
          JSR PRT
          LDA $FD
          JSR PRT
          LDC #$00
          LDA ($FD,X)
          JSR SPCPRT
          JMP ($0334)   ;POWROT Z
                        ;PRZERWANIA
;-------------------------
SPCPRT                  ;"-" + PRINT BYTE
          PHA
          LDA #$2D
          JSR LB
          PLA
PRT                     ;PRINT BYTE
          PHA
          LSR A
          LSR A
          LSR A
          LSR A
          JSR LD
          PLA
LD        AND #$0F
          CMP #$0A
          BCC BL
          SBC #$09
          BNE LB
BL        ORA #$30
LB        STA $0400,Y
          INY
          RTS
;-------------------------
```

JAK POŁĄCZYĆ DWA C-64?

Po opublikowaniu schematu przewodu łączącego dwie Amigi poprzez łącze szeregowe RS-232 do redakcji zaczęły napływać listy z pytaniami o możliwość i ewentualny schemat połączenia za sobą dwóch C-64 oraz C-64 z Amigą w analogiczny sposób. Oczywiście oba połączenia są możliwe.

Do wykonania połączenia potrzebne są: dwie wtyczki do gniazda USER PORT C-64 oraz trójżyłowy przewód. Schemat kabla przedstawiony jest na rysunku.

Po wykonaniu podłączamy go, oczywiście do WYŁĄCZONYCH komputerów. Aby dokonać transmisji, należy posłużyć się urządzeniem numer 2, będącym niczym innym jak portem RS-232C. Przykładowe programy w BASIC-u V2.0 wyglądają następująco:

1) dla nadajnika:
```
10 OPEN 1,2,0,CHR$(8)+CHR$(0)
20 GET K$:IF K$="" THEN 20
30 PRINT#2,K$;:PRINT K$;
40 GOTO 20
```

2) dla odbiornika:
```
10 OPEN 1,2,0,CHR$(8)+CHR$(0)
20 GET#2,R$:IF R$="" THEN 20
30 PRINT R$;
40 GOTO 20
```

Oba programy otwierają kanał komunikacyjny dla urządzenia o numerze 2 czyli RS-232C. Dwa znaki określają rodzaj transmisji: pierwszy - długość słowa, liczbę bitów stopu i szybkość przesyłania informacji, drugi - sposób kontroli parzystości, rodzaj pracy i rodzaj potwierdzenia gotowości. Znaczenie poszczególnych bitów przedstawiam poniżej.

Reszta programu nie wymaga chyba komentarza. Oczywiście dla ułatwienia transmisji w obie strony można napisać znacznie bardziej rozbudowany program.

Uwagi:

1. Ze względu na inne poziomy napięć niż w standardowym RS-232, długość przewodu jest ograniczona do kilku, najwyżej kilkunastu metrów.
2. Aby transmisja przebiegała poprawnie, na obu komputerach muszą być ustawione identyczne parametry transmisji.
3. Opisanym kablem można połączyć także dwa C-128 oraz C-64 z C-128.

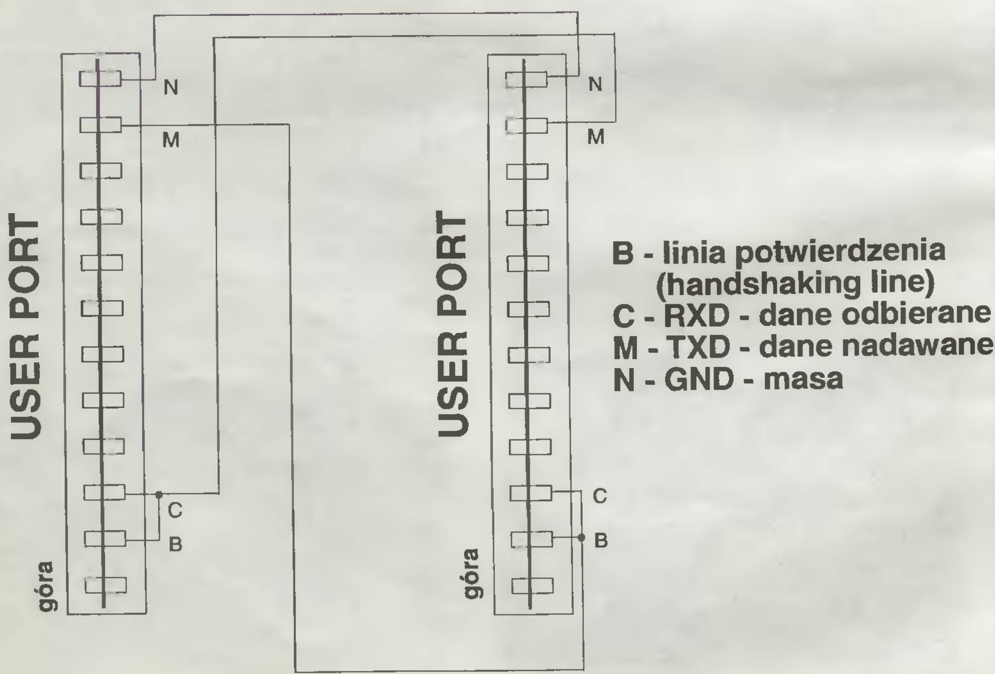
JERZY DUDEK

Pierwszy parametr:

Nr bitu	Funkcja	Wartości
7,6,5	Kontrola parzystości	000 - brak, 001 - nieparzystość, 011 - parzystość
4	Dupleks	0 - pełny dupleks, 1 - półdupleks
3,2,1	Nie używane	
0	Rodzaj łącza	0 - 3 linie, 1 - X linii

Drugi parametr:

Nr bitu	Funkcja	Wartości
7	Liczba bitów	0 - 1 bit, 1 - 2 bity
6 i 5	Długość słowa	00 - 8 bitów, 01 - 7 bitów 10 - 6 bitów, 11 - 5 bitów
4	Nie używany	
3,2, 1,0	Prędkość transmisji	0001 - 50 bodów 0010 - 75 bodów 0011 - 110 bodów 0100 - 134,5 bodów 0101 - 150 bodów 0110 - 300 bodów 0111 - 600 bodów 1000 - 1200 bodów 1010 - 2400 bodów



SUPERMARKET

Co jest grane !?

Celem usprawnienia funkcjonowania rubryki "SUPERMARKET" wprowadziliśmy nowe zasady umieszczania ogłoszeń. Oto one:

1. Ogłoszenie może zamieścić każdy Czytelnik "Commodore & Amiga".
2. Ogłoszenie jest bezpłatne.
3. Ogłoszenia do "SUPERMARKETU" przyjmowane są tylko od osób prywatnych.
4. Wszystkie ogłoszenia traktowane są jako jednorazowe bez względu na liczbę nadesłanych kuponów. Jeżeli Czytelnik chce, aby jego oferta została wielokrotnie opublikowana, musi nadesłać do redakcji oddzielne listy z kuponami.
5. Ogłoszenia dotyczące sprzedaży, kupna i wymiany nielegalnych kopii programów (gry, użytki) oraz nielegalnych kopii książek będą automatycznie odrzucane.
6. Ogłoszenie, oprócz oferty, musi zawierać dokładny adres ogłaszającego tzn. imię, nazwisko, miasto, kod pocztowy, nazwę ulicy, numer domu, numer mieszkania - wszystko wypisane drukowanymi literami.
7. Koperta z ogłoszeniem musi mieć wyraźny dopisek SUPERMARKET, oferta bez dopisku nie będzie drukowana.

Ogłoszenia będą ukszywać się w kolejności nadsyłania.

REDAKCJA



↓ wytnij i naklej na kopertę ↓

SUPERMARKET

Redakcja
COMMODORE & AMIGA
02-776 Warszawa
ul. Wasilkowskiego 7

COMMODORE 64/126

■ Sprzedam C-64VGS, magnetofon, X, VGS, mysz, 2 joysticki, pokrywa, przedłużacz do joysticka, 19 kaset z grami. Cena około 2,4 mln zł. Dariusz Oktaba, 02-795 Warszawa, ul. Kazury 5/41.

■ Zamienię C-64, magnetofon, Black Box, TV-monitor (Neptun 150 E), 2 joysticki, 300 gier i programów, dopłata 1,5 mln zł. na Amigę 500+ lub Amigę 500 z modulatorem. Remigiusz Rynkiewicz, 37-300 Leżajsk, ul. T. Michałka 35.

■ Sprzedam C-64II, stację 1541II, magnetofon 1530 (wszystko na gwarancji) Final II, Black Box, dwa joysticki, 14 kaset (około 300 gier i użytkowych), 14 dyskietek, pudełko na 80 dyskietek literaturę. Cena 4 mln zł. Sławomir Piątek, 64-000 Kościan, ul. Kołłątaja 3.

■ Sprzedam C-64II, stację 1541II (wszystko na gwarancji), magnetofon 1530, Final III, 2 joysticki, pudełka na dyskieteki, literaturę. Bartosz Pasternak, 59- 975 Sulików, ul. Mikułowa 43.

■ Sprzedam C-64, magnetofon, joystick, Black Box, 2 moduły z grami, literaturę, 50 kaset z programami. Cena 2,5 mln zł. Tadeusz Piędel, 32-510 Jaworzno, ul. Olszewskiego 2c/46.

■ Sprzedam C-64 stan idealny, magnetofon, joystick, Final III, Turbo Master, mysz z mouse pad, oprogramowanie. Janusz Hassa, 44-164 Gliwice-Brzezinka, ul. Zakopiańska 3.

■ Sprzedam C-64, magnetofon 1530, stację 9900, joystick, Final III, moduł z grami, 20 dysków, 10 kaset, pudełko na 100 dyskietek, literaturę. Cena 2,5 mln zł. Krzysztof Wardaszko, 05-200 Wołomin, ul. Kobyłkowska 30/6, tel. 76-40-19.

■ Sprzedam C-64 (gwarancja), magnetofon (gwarancja), monitor zielony Unitra Unimor, moduł X, pokrywę, około 50 programów na kasetach oraz "Podręcznik użytkownika C-64". Cena całości 2,9 mln zł. K. Andruszczak, 03-343 Warszawa, ul. Rembielińska 3/153.

■ Sprzedam C-64II, magnetofon, Black Box 4.0 i egzemplarze "C&A" 1-8 lub całość wymienię na Atari 65 XE z podobnym osprzętem. Dariusz Włoch, 59-700 Bolesławiec, ul. Staroszkolna 4b/3.

■ Zamienię C-64II, magnetofon TURBO, 2 moduły, 2 joysticki, mysz GEOS, 2 książki, około 300 użytków i gier. Wszystko wymienię na używaną Amigę 500 z ewentualną dopłatą. Piotr Duszyński, 80-461 Gdańsk, ul. Startowa 71c m 26, tel. 56-67-10.

■ Sprzedam C-64 (na gwarancji), magnetofon, rozbudowany X, literatura i inne. Tomasz Czubaty, Wrocław, ul. Szybowcowa 14/1, tel. 51-44-93.

■ Sprzedam roczny C-64, magnetofon, 2 joysticki, Blue Box IV, 6 kaset z programami, cartridge, literatura. Tomasz Czarnecki, Obłęgor 116, 26-067 Strawczyn.

■ Sprzedam C-64II (stan idealny), magnetofon, TOP STAR, Final III, 40 kaset z grami i użytkami, literaturę. Cena całości: 2,3 mln zł. Tomasz Szatewicz, 11- 700 Mrągowo, Oś. Mazurskie 18/17.

■ Sprzedam Commodore C-64II, magnetofon 1530 C2N, drukarkę (pełne wyposażenie), moduły Black Box i Final III, joystick, pokrywę na komputer, 18 kaset i literaturę. Cena około 5 mln zł. Robert Reterski, 44-100 Gliwice, ul. Saturna 3/12.

INNE

■ Sprzedam: kolorowy telewizor-monitor Samsung 14 cali (z pilotem), posiadający wejścia cinch na sygnały audio i video. Cena 2,9 mln zł. Również monitor Neptun czarno-biały 12 cali, cena 850 tys. zł oraz "Bajtki" z lat 1989/90/91, "Top Secret" za 50% aktualnej ceny. Artur Olszański, 20-323 Lublin, ul. Puchacza 9/12 tel. 419-42.

■ Kupię mało używany kolorowy monitor 1802 lub inny monitor Commodore do C-64. Piotr Grubek, 05-230 Kobyłka, Kobylak, ul. Dąbrowskiego 12a.

■ Sprzedam dwa moduły do C-64: Final III (250 tys. zł), moduł z grami (150 tys. zł), dwa joysticki (uszkodzone) + części do naprawy (100 tys. zł). Za przesyłkę płaci odbiorca. Rafał Chapiewski, 89-600 Chojnice, ul. Waryńskiego 4a.

■ Kupię moduł z interpreterm SIMONS BASIC, literaturę na temat tego języka. Nawiążę również kontakt z uczącymi się tego dialektu BASIC-a. Piotr Oronowski, 00- 650 Chynów, Sułkowice 94, woj. radomskie.

■ Sprzedam moduły do C-64: częstościomierz 30 Mhz - 100 tys. zł., port 24 linie TTL wejścia/wyjścia - 150 tys. zł. Jarosław Piaszczyński, 63-211 Golina, ul. Jarocińska 27.

■ Sprzedam drukarkę SEIKOSHA GP 500 VC II do C-64 wraz z 1000 arkuszy papieru oraz z kasetą z edytorami textu. Cena całości - 1,3 mln zł. Informację koperta zwrotna + znaczek. Edyta Łabiak, 35-213 Rzeszów, ul. Ofiar Katynia 8/194.

■ Sprzedam drukarkę Commodore MPS 802 do C-64/C-128 oraz Final II. Cena 700 tys. zł. Kazimierz Łysiak, 39-400 Tarnobrzeg, ul. Kopernika 7/15, tel. 22-45-27.

■ Kupię drukarkę lub stację dysków do C-64. Arkadiusz Lewandowski, 64-306 Boruja Kościelna, ul. Ogrodowa 14.

■ Poszukuję kontaktów w sprawie redefinicji polskich znaków dla drukarki STAR LC 200 Color (dla AMIGI) lub EPROMU z polskimi znakami też do LC 200. G. Dobrowolski, 31-102 Kraków, ul. Zwierzyniecka 14/8, tel. (0-12) 21-61-52.

■ Kupię książkę J. Ruszczyca "Poznajemy FORTH". Nawiążę kontakt z programującymi w 6510. M. Kędzierski, 59-257 Gromadka, ul. Szkolna 20b/5.

■ Sprzedam książki: "Wprowadzenie do Amiga DOS" - 80 tys. zł., "Motorola 68000" - 70 tys. zł., "BASIC cz. 2 komendy" - 60 tys. zł., "Tips & Tricks cz. 1 i cz. 2" - po 50 tys. zł. za sztukę. Krzysztof Szymkowicz, 89-100 Nakło n. Notecią, ul. K. Wielkiego 8/15, tel. 85-20-10.

■ Kupię pilnie książkę "Der Commodore 64 und der Rest der Welt" Brueckmanna. Piotr Wyrzykowski,- 15-836 Białystok, ul. Ukośna 1/13.

■ Poszukuję książek: "Jak rozbudować interpreter" K. Gajewski, B. Radziszewski; "Commodore C-64" B. Frelek oraz programów Macroasembler, Turboassembler i jakiegokolwiek programu monitora (kaseta C-64). Jarosław Kaczmarek, 64-400 Międzychód, ul. G. Sikorskiego 23c/1.

■ Poszukuje dobrej mapy pamięci C-64. Schematu i oprogramowania samplera do C-64 (magnetofon). Jędrzej Chmielewski, 85-611 Bydgoszcz, ul. Kąkolowa 7/33.



World of Commodore

Firma Commodore rozrasta się ostatnimi laty, nic więc dziwnego, że przestaje jej wystarczać uczestniczenie w targach organizowanych przez innych i pragnie mieć coraz więcej własnych. Do corocznych AmiExpo w Kolonii dołączyły więc World of Commodore we Frankfurcie.

Targi World of Commodore miały w tym roku swoją premierę, są bowiem imprezą zupełnie nową. Od AmiExpo różnią się pod kilkoma względami: po pierwsze są, jak na razie, niezbyt wielkie, po drugie zaś - poświęcone są wszystkim produktom firmy Commodore, a więc nie tylko Amidze, ale również i pecetom.

Mówiąc w dużym skrócie, WOC '92 nie były imprezą olbrzymią. Nie były nawet imprezą dużą. Nie ma tu się jednak czemu dziwić, w końcu targi te odbywały się w tym roku po raz pierwszy. Co do ich podziału pomiędzy poszczególne komputery produkowane przez Commodore, to śmiało można powiedzieć, że targi zdominowała Amiga. Było nieco pecetów, trudno było natomiast uświadczyc Commodore 64. Całość podzielona była oprócz tego na dwie części: jedną, dla fanów gier komputerowych, i drugą - tzw. profi - poświęconą profesjonalnym zastosowaniom Amigi i pecetów.

Nie zaprezentowano co prawda zbyt wielkiej liczby nowości, trochę ich jednak było. Nie będę ich tu oczywiście wszystkich omawiał, napomykając o nich zachęcam Was bowiem do sięgnięcia za miesiąc po następny numer C&A, w którym znajdzie się reportaż z targów. Z pewnością jednak warto już teraz wspomnieć o najnowszym produkcie firmy Commodore, a mianowicie o Amidze 1200. Jest to, jak łatwo się domyśleć, "udomowiona" wersja Amigi 4000, a więc komputer tani, ale wykorzystujący potężne procesory specjalizowane A4000.

Tak więc A1200 ma IDENTYCZNE możliwości graficzne jak A4000, co oznacza ni mniej ni więcej, że bije pod tym względem na głowę wszystkie starsze Amigi (nie wyłączając A3000, oczywiście), a także wszystkie pecety i Atari razem wzięte. Nowa Amiga oferuje bowiem paletę 16.8 mln. kolorów i możliwość równoczesnego uzyskania na ekranie 256 tysięcy z nich. Maksymalną liczbę kolorów można wyświetlić także przy najwyższej rozdzielczości, a więc 1280x512. Powtarzam wszystkie te znane z Amigi 4000 dane, by uświadomić Wam, jak potężne możliwości graficzne ma ten tani (kosztujący tylko 765 DM po odliczeniu 14% niemieckiego VAT!) komputer domowy.

Nowe możliwości graficzne to oczywiście nie wszystkie usprawnienia, jakie wprowadzono w A1200. Jednym z najważniejszych jest zastosowanie nowego procesora, Motoroli 68EC020, taktowanego częstotliwością 14 MHz. Tak więc A1200 jest maszyną w pełni 32-bitową i szybkością pracy kilkukrotnie przewyższa Amigi 500 i 600. Firma Commodore przyjęła też wreszcie do wiadomości, że czasy, gdy 1MB wystarczał do wszelkich zastosowań, dawno minęły i instaluje w A1200 2MB pamięci. Możliwa jest wewnętrzna rozbudowa do 4MB, po wykorzystaniu gniazda PCMCIA można jeszcze "dorzucić" dodatkowe 4MB. Wspomniane gniazdo PCMCIA spełnia taką samą rolę jak w przypadku A600, w Amidze 1200 nie jest ono jednak jedynym złączem dającym bezpośredni dostęp do systemu - dodano bowiem pełną, 32-bitową szynę procesora, dzięki czemu A1200 ma takie same możliwości rozbudowy jak A500.

Podobnie jak A600, w modelu 1200 wbudowany jest kontroler dysku twardego standardu AT-BUS, wystarczy więc kupić sam napęd. Jest też niezbędne w komputerze czysto domowym kolorowe wyjście Composite Video oraz modulator.

I to na razie tyle, powyższa notatka nie jest bowiem sprawozdaniem z targów, jej zadaniem było jedynie poinformowanie Was o nich i o nowej Amidze. Zapraszam więc do lektury bardzo obszernego, wspomnianego wyżej, reportażu z targów w następnym numerze C&A. Znajdziecie tam również więcej wiadomości o Amidze 1200, którą już teraz wszystkim polecam.

Andrzej Bobek

Dane techniczne Amigi 1200

CPU (główny procesor):
- Motorola 68EC020, 14MHz
- brak koprocesora matematycznego i MMU

PAMIĘĆ:
- 2MB CHIP-RAM, nierozszerzalna
- 0MB FAST-RAM, rozszerzalna wewnętrznie do 2MB, a przy użyciu złącza PCMCIA - do 6MB

SYSTEM OPERACYJNY:
- 512KB, ROM, Kickstart 3.0, prawie identyczny jak A4000

ZŁĄCZA:
- 2 * Mysz/Joystick
- szeregowe (RS-232)
- równoległe (Centronics)
- monitora (RGB lub RGBI)
- stacji dysków
- wewnętrzny port dla napędów AT-BUS
- dwóch kanałów stereo
- Composite Video
- wyjście modulatora

ZŁĄCZA SYSTEMU:
- identyczne jak w A600, standard PCMCIA
- szyna procesora umieszczona pod klawiaturą, zapewniająca duże możliwości rozbudowy

KLAWIATURA:
- stanowi całość z komputerem
- 94 klawisze, wydzielony blok numeryczny - w zasadzie identyczna jak w A500

MYSZ:
- optyczno-mechaniczna, dwa przyciski, identyczna jak w A4000

STACJA DYSKÓW:
- wbudowana zwykła stacja 3,5 cala
- pojemność 880KB

TWARDY DYSK:
- wbudowany interface AT-BUS, możliwość wewnętrznego podłączenia dowolnego napędu 2.5-calowego w tym standardzie

GRAFIKA:
- nowy zestaw kości specjalizowanych AGA (nazywany też AA), identyczny jak w Amidze 4000
- rozdzielczości od 320x256 do 1280x512 (nie licząc obszaru ramki)
- zmienna częstotliwość pracy kości graficznej
- 24-bitowa paleta kolorów, 16.777.216 możliwości
- 2 do 262.144 kolorów na raz na ekranie

TRYBY PRACY WYJŚĆ DLA MONITORA:
- zwykły monitor RGB lub VGA (niektóre tryby graficzne dostępne tylko z monitorem VGA)
- częstotliwość pozioma: 15-31 kHz
- częstotliwość ramki: 50-72 Hz

DŹWIĘK:
- cztery 8-bitowe przetworniki, po dwa na kanał stereo